Periodicità mensile • OTTOBRE 2017 • ANNO XXI - N.9 (218) - 13/09/2017
PRESS-DI DISTRIBUZIONE STAMPA E MULTIMEDIA S.R.L. - 20090 SEGRATE

VERSIONE PLUS RIVISTA+LIBRO+CD € 9,99
VERSIONE STANDARD RIVISTA+CD € 6,99

ioProgrammo

PER ESPERTI E PRINCIPIANTI

Poste Italiane S.p.A Spedizione in A.P. - D.L. 353/2003 (CONV.IN L.27/02/2004 N.46) ART.1 COMMA1 DCB ROMA

GRATIS SOLO PER TE!

VISUAL STUDIO PER ARDUINO

Impara a **realizzare gli sketch direttamente nel tuo IDE preferito**. Con **Visual Basic**, il plug-in giusto e la nostra guida pratica, l'elettronica diventa un gioco

ALL'INTERNO TUTTO IL CODICE E IL PLUG-IN PRONTO DA INSTALLARE

ANTEPRIMA ESCLUSIVA

AMAZON ECHO
Con Raspberry Pi lo fai da te!

La guida per "clonare" lo smart speaker e implementare **nuove skill** su **Alexa** usando**JSON** come interfaccia

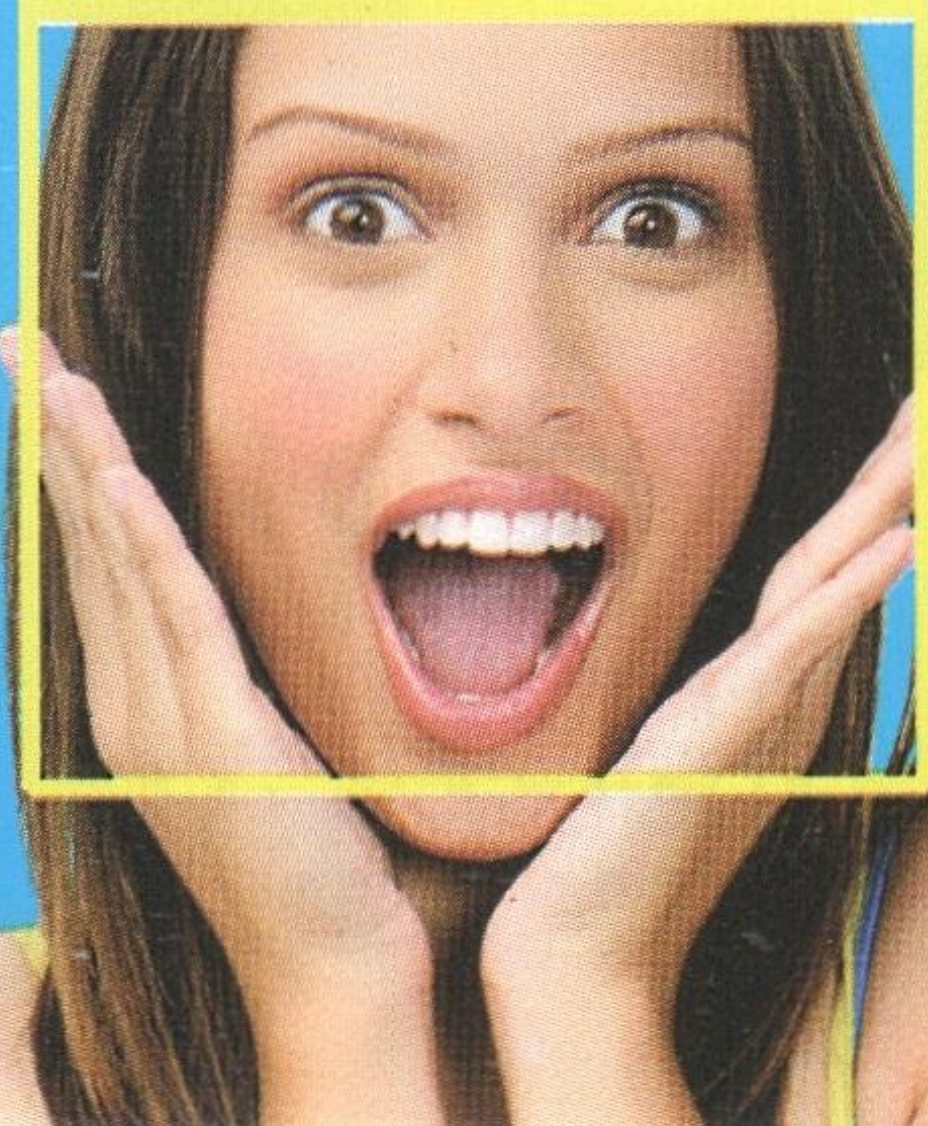
Ed in più... se non vuoi programmare, addestri Alexa con IFTTT!

WEB 2.0
METTI GOOGLE CALENDAR IN WORDPRESS
Realizziamo un widget per WP che si sincronizza con il calendario di Big G

INTELLIGENZA ARTIFICIALE
COM'È UMANO IL BOT!
Una chat automatica che risponde agli utenti in linguaggio "naturale"

MULTIPIATTAFORMA
XAMARIN VS CORDOVA: SCONTRO AL VERTICE
Vuoi che le tue App girino su più device? Scopri la piattaforma più adatta alle tue esigenze

SICUREZZA
I PERICOLI DEL DNS!
Attacchi Spoofing: come si realizzano e come ci si difende

MICROSOFT COMPUTER VISION API
LA NOSTRA APP IMPARA A VEDERE!
Dai sistemi di sicurezza alla scansione di magazzino: scopriamo come addestrare un'app a riconoscere volti, simboli e oggetti

50+ SNIPETS PRONTI ALL'USO
per C#, Java, Python, PHP, iOS, Android...

Anno XXI - N.ro 9 (218) - Ottobre 2017
Periodicità Mensile - P.i. 13/09/2017
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale: Massimo Mattone
Direttore Responsabile: Massimo Mattone
Responsabile Editoriale: Gianmarco Bruni
Collaboratori: Carlo Daniele, Mario De Ghetto, Sebastiano Galazzo, Vito Ganci, Francesco Napoletano, Luca Tringali
Coordinamento redazionale: Raffaele del Monaco
Redazione: Giancarlo Giovinazzo
Segreteria di Redazione: Rossana Scarcelli

Realizzazione grafica: Cromatika S.r.l.
Responsabile produzione: Giancarlo Sicilia
Responsabile grafico di progetto: Salvatore Vuono
Illustrazioni: Arturo Barbuto, Tony Intieri
Grafica: Fabiola Grandinetti, Lino Pelle, Luigi Ferraro, Elio Monaco

Concessionaria per la pubblicità: EMOTIONAL PUBBLICITÀ Srl
Via F. Melzi d'Eril, 29
20154 Milano - Tel 02/76318838
e-mail advertising@edmaster.it

Editore: Edizioni Master S.p.a.
Via B. Diaz, 13 - 87036 Rende (CS)
Presidente e Amministratore Delegato: Massimo Sesti

Abbonamento e arretrati
ioProgrammo (6 numeri) € 26,00 sconto 38% sul prezzo di copertina di € 41,94
• ioProgrammo con Libro (6 numeri) € 36,99 sconto 38% sul prezzo di copertina di € 59,94
ioProgrammo (12 numeri) € 49,99 sconto 40% sul prezzo di copertina di € 83,88
• ioProgrammo con Libro (12 numeri) € 72,00 sconto 40% sul prezzo di copertina di € 119,88
Offerte valide solo per l'Italia fino al 30/11/2017.
Costo arretrati (a copia): il prezzo di copertina + € 6,10 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate inviando una e-mail all'indirizzo arretrati@edmaster.it
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 199.50.00.05*, oppure via posta a EDIZIONI MASTER Via Diaz, 13 - 87036 Rende (CS), dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito Visa, Cartasì, o Eurocard/Mastercard (inviando la Vs. autorizzazione, il numero di carta di credito, la data di scadenza, l'intestatario della carta e il codice CVV2, cioè le ultime 3 cifre del codice numerico riportato sul retro della carta).
- bonifico bancario intestato a Edizioni Master S.p.A. c/o BANCA DI CREDITO COOPERATIVO DI CARUGATE E INZAGO S.C. IBAN IT47R0845333200000000066000 (inviando copia della distinta insieme alla richiesta)

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.

Sostituzioni: qualora nei prodotti fossero rinvenuti difetti o imperfezioni che ne limitassero la fruizione da parte dell'utente, è prevista la sostituzione gratuita, previo invio del materiale difettato. La sostituzione sarà effettuata se il problema sarà riscontrato e segnalato entro e non oltre 10 giorni dalla data effettiva di acquisto in edicola e nei punti vendita autorizzati, facendo fede il timbro postale di restituzione del materiale.
Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master - Servizio Clienti - Via Diaz, 13 - 87036 Rende (CS)
Assistenza tecnica: ioprogrammo@edmaster.it

SERVIZIO CLIENTI
@ serviziolienti@edmaster.it
199.50.00.05* sempre in funzione
199.50.50.51* dal lunedì al venerdì 10.00 – 13.00
*Costo massimo della telefonata 0,118 € + iva a minuto di conversazione, da rete fissa, indipendentemente dalla distanza. Da rete mobile costo dipendente dall'operatore utilizzato.

Stampa: Arti Grafiche Boccia S.p.A. - Via T. C. Felice, 7 - 84131 Salerno
Duplicazione CD-Rom: EcoDisk S.r.l. - Via Enrico Fermi, 13 Burago di Molgora (MB)
Distributore esclusivo per l'Italia:
Press-di Distribuzione Stampa e Multimedia S.r.l. - 20090 Segrate

Finito di stampare nel mese di Settembre 2017

"Rispettare l'uomo e l'ambiente in cui esso vive e lavora è una parte di tutto ciò che facciamo e di ogni decisione che prendiamo per assicurare che le nostre operazioni siano basate sul continuo miglioramento delle performance ambientali e sulla prevenzione dell'inquinamento"

▼LO SVILUPPO SOFTWARE È A UNA SVOLTA

Grazie alla solerzia dei nostri collaboratori, anche questo mese potete trovare su ioProgrammo grandi anticipazioni tecnologiche. Per tutte, basti pensare alla programmazione dell'assistente virtuale di Amazon: negli Stati Uniti è ormai una sorta di standard de facto. La capacità di istruire Alexa dovrebbe far parte del bagaglio di ogni programmatore e non solo per l'utilità del servizio in sé, quanto per le prospettive che schiude e per i nuovi paradigmi di interazione imposti dai nuovi dispositivi. Ripensare le applicazioni che sviluppiamo in ragione di una interfaccia puramente vocale richiede una radicale rivisitazione delle consuetudini progettuali. Quindi, l'invito è di sperimentare fin da subito, anche se Amazon Echo non è ancora disponibile in Italia: in questo modo potremo assimilare fin da subito questo nuovo approccio e non ci faremo trovare impreparati all'onda lunga che sta arrivando dall'America. A circa venti anni di distanza dalla nascita di questa rivista, stiamo assistendo alla prima vera grande rivoluzione nel mondo della programmazione: Intelligenza Artificiale, realtà aumentata, interfacce vocali… tecnologie la cui evoluzione abbiamo seguito da vicino e che ora, finalmente mature, sembrano spingere tutte assieme verso una singolarità.

Raffaele del Monaco

All'inizio di ogni articolo, troverete un simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre \soft\codice\ e \soft\tools\) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it

VISUAL STUDIO PER ARDUINO

Impara a realizzare gli sketch direttamente nel tuo IDE preferito. Con Visual Basic, il plug-in giusto e la nostra guida pratica, l'elettronica diventa un gioco

MICROSOFT COMPUTER VISION API

LA NOSTRA APP IMPARA A VEDERE!

Dai sistemi di sicurezza alla scansione di magazzino: scopriamo come addestrare un'app a riconoscere volti, simboli e oggetti

Costruire algoritmi per generare conoscenza

Elaborare il magma di dati oggi disponibile è una sfida affascinante e imprescindibile per il mondo contemporaneo dove la conoscenza e l'informazione sono il primo valore. Il machine learning è la risposta: grazie ai suoi algoritmi è possibile creare macchine in grado apprendere in maniera automatica e rispondere alle domande chiave per il successo. Questo libro accompagna nel mondo del machine learning e mostra come Python sia il linguaggio di programmazione ideale per costruire algoritmi sofisticati in grado di interrogare i dati nel modo migliore e recuperare preziosi insight. Viene spiegato l'utilizzo di librerie Python dedicate – tra cui scikit-learn, Theano e Keras – applicate in ambiti come la selezione e la compressione dei dati, l'analisi del linguaggio naturale, l'elaborazione di previsioni, il riconoscimento delle immagini. L'approccio didattico è pragmatico: tutti i concetti sono accompagnati da esempi pratici di codice. La lettura è consigliata a chi ha già alle spalle qualche studio teorico nel campo del machine learning oltre a una buona conoscenza di programmazione in Python.

ARGOMENTI TRATTATI

- ✓ Addestrare gli algoritmi a compiti di classificazione
- ✓ Utilizzare i classificatori della libreria scikit-learn
- ✓ Selezionare i dati con le tecniche di pre-elaborazione
- ✓ Ridurre la dimensionalità dei dati con le tecniche compressione
- ✓ Conoscere, valutare e combinare i modelli di machine learning
- ✓ Creare analisi del sentiment elaborando dati testuali dai social media
- ✓ Integrare modelli di machine learning in applicazioni web
- ✓ Elaborare previsioni attraverso i modelli di analisi a regressione
- ✓ Individuare nuovi cluster e pattern di dati
- ✓ Addestrare reti neurali al riconoscimento delle immagini
- ✓ Lavorare con Theano per ottimizzare gli algoritmi di machine learning

Come usare l'interfaccia del CD-ROM

NON PERDERE I NOSTRI **MAXI SCONTI!**

SCONTI FINO AL 56%

ABBONATI SUBITO

E IN PIÙ SCOPRI COME RICEVERE LE NOSTRE IMPERDIBILI RIVISTE, COMODAMENTE A CASA TUA CON UNO **SCONTO SPECIALE!**

WIN MAGAZINE
È la rivista di informatica e tecnologia più venduta in Italia, ricca di idee e progetti, spiegati con un linguaggio chiaro e accessibile. Contenuti pratici e capaci di stimolare un utilizzo creativo del computer e della tecnologia.

TURISTI PER CASO MAGAZINE
È la prima rivista scritta dai turisti per i turisti e propone percorsi, suggerimenti ed emozioni direttamente dalla voce dei viaggiatori. Offre una vasta gamma di informazioni utili e consigli indispensabili. Ha come testimonial e autori d'eccezione Syusy Blady e Patrizio Roversi.

PLAY GENERATION
È il mensile dedicato agli appassionati del mondo PlayStation, di tutte le età. All'interno ampio spazio a tutto l'universo PlayStation: news, anteprime, recensioni, trucchi e prove comparative.

Ritaglia e spedisci il coupon in busta chiusa a: **EDIZIONI MASTER S.p.A.** Via Diaz, 13 - 87036 Rende (CS) oppure invialo via fax al n. 199.50.00.05 o vai sul sito **http://abbonamenti.edmaster.it**

Si, desidero abbonarmi a **ioProgrammo**

- ☐ base 6 Numeri € 24,99 anziché € 41,94
- ☐ base 12 Numeri € 41,99 anziché € 83,88
- ☐ con libro 6 Numeri € 33,99 anziché € 59,94
- ☐ con libro 12 Numeri € 52,99 anziché € 119,88
- ☐ Win Magazine base 12 Numeri € 24,99 anziché € 35,88
- ☐ Turisti per Caso 22 Numeri € 27,99 anziché € 48,40
- ☐ Play Generation 11 Numeri € 24,99 anziché € 32,89

Informativa ex art. 13 d.lgs. 196/2003 "codice in materia di protezione dei dati personali": Edizioni Master Spa con sede in Rende, c.da Lecco n. 64 - Z. Ind. - , in qualità di "Titolare" del trattamento, è tenuta a fornirLe alcune informazioni su l'utilizzo dei Suoi dati personali. I dati personali raccolti da Edizioni Master, nel presente coupon, sono conferiti direttamente dall'interessato e sono trattati, indispensabilmente, al solo fine di dare esecuzione alla Sua richiesta di abbonamento; per tale finalità non è richiesto il consenso, ex art.24 comma 1 lett. b. I trattamenti saranno effettuati mediante strumenti manuali, informatici e telematici, con logiche correlate al rapporto in essere ed agli obblighi previsti dalle leggi vigenti. L'interessato potrà esercitare, presso la Edizioni Master Spa, i diritti di cui all'art. 7 del D.Lgs. 196/2003: modifica, cancellazione, correzione, etc. I dati ravccolti, potranno essere comunicati, per la stessa finalità, ai Responsabili ed agli Incaricati designati da Edizioni Master, owero a società collegate e controllate, facenti parte del medesimo gruppo editoriale; potranno altresì essere trattati per finalità promo-pubblicitaria, commerciale, per l'invio di altre offerte, per indagini di mercato con il suo consenso esplicito.

COMPILARE CON I PROPRI DATI ANAGRAFICI

Nome ____
Cognome ____
Via ____
n° ____ CAP ____ Provincia ____
Città ____
n. cellulare ____
n. telefono ____
e-mail ____

REGALO L'ABBONAMENTO A (compilare solo in caso di abbonamento regalo)

Nome ____
Cognome ____
Via ____
n° ____ CAP ____ Provincia ____
Città ____
n. cellulare ____
n. telefono ____
e-mail ____

sesso: ____ data di nascita: ____ stato civile: ☐ Celibe/Nubile ☐ Coniugato ☐ Vedovo/a ☐ Divorziato/a ☐ Convivente

titolo di studio: ☐ Lic. Scuola Elementare ☐ Lic. Scuola Media Inferiore ☐ Diploma ☐ Laurea professione: ____

Dichiaro di essere maggiorenne e autorizzo il trattamento dei miei dati personali per le finalità indicate nell'Informativa ☐ SI ☐ NO

Scelgo di effettuare il pagamento attraverso:

☐ Bonifico bancario intestato a EDIZIONI MASTER S.p.A.
BANCA DI CREDITO COOPERATIVO DI CARUGATE E INZAGO S.C.
IBAN IT47R0845333200000000066000
(inviando copia della distinta via fax oppure via posta)

☐ Assegno bancario non trasferibile intestato ad EDIZIONI MASTER S.p.A. (Allegato in busta chiusa)

Carta di credito ○ VISA ○ Cartasì ○ Mastercard

n. ____
(riporta il numero completo della carta indicandone tutte le cifre)

____ (scadenza) ____ CVV2* *ultime 3 cifre del codice numerico riportato sul retro della carta

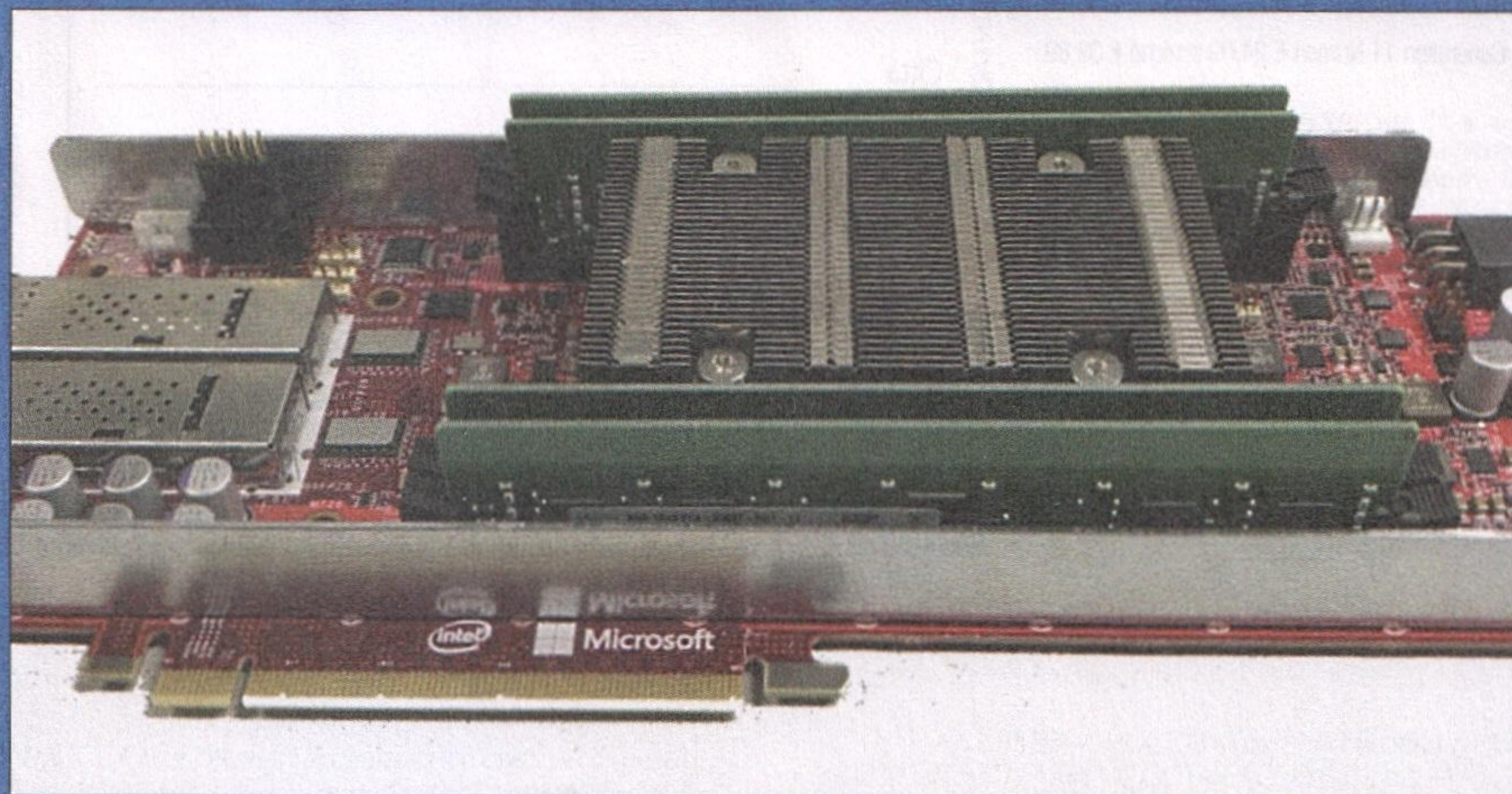
Firma ____

ioProgrammo 218 - offerta valida fino al 31/10/2017

Android 8.0 - Oreo è ricco di novità

A poco più di un anno dal debutto iniziale, Android "O" è ufficialmente diventato "Oreo": l'ottava release del sistema operativo mobile più usato al mondo sfrutta una partnership commerciale a tema dolciario come già successo con KitKat (4.4), ed è infarcito con una serie di novità architetturali significative oltre alle solite funzionalità innovative visibili all'utente finale. Come da tradizione, Google presenta Android 8.0 Oreo come un OS più veloce, "intelligente", potente e dolce che mai, una piattaforma che sarà soprattutto una novità per i consumatori mainstream visto che in questi mesi la corporation dell'advertising non ha mancato di distribuire le versioni preliminari del sistema a beneficio degli sviluppatori di app. Per quanto riguarda le performance, Android Oreo introduce un controllo più stringente sulle app che lavorano in background così da risparmiare sia i cicli-processore che la batteria; migliorata anche la sicurezza dell'OS grazie - tra l'altro - alla già presentata funzionalità Play Protect, mentre le ottimizzazioni alla funzionalità di garbage collection e all'efficienza nell'utilizzo della memoria dovrebbero garantire tempi di avvio del terminale sensibilmente inferiori. Anche il centro notifiche è stato rivisto con le notifiche delle app inserite in un canale separato, mentre 60 emoji risultano riprogettate per il piacere di chi non aspetta altro che inserire buffe faccine nelle sue sessioni di chat con amici e parentado assortito. Oreo segna poi il debutto del Progetto Treble, una tecnologia ideata per separare il software e le app di terze parti dal framework proprio di Android così da favorire l'accelerazione della distribuzione delle nuove release. O quantomeno per invogliare i produttori OEM a rilasciare gli update/upgrade più velocemente. E parlando di produttori OEM, Google rassicura gli utenti sul fatto che tutti i principali partner del progetto Android (Samsung, HTC, Huawei, Motorola, Sharp, Sony e altri) dovrebbero lanciare o aggiornare i "nuovi" terminali con l'ultima release di Android. Per i dispositivi mobile commercializzati direttamente da Google delle linee Nexus e Pixel, invece, sono già disponibili le immagini di sistema da installare manualmente nel firmware.

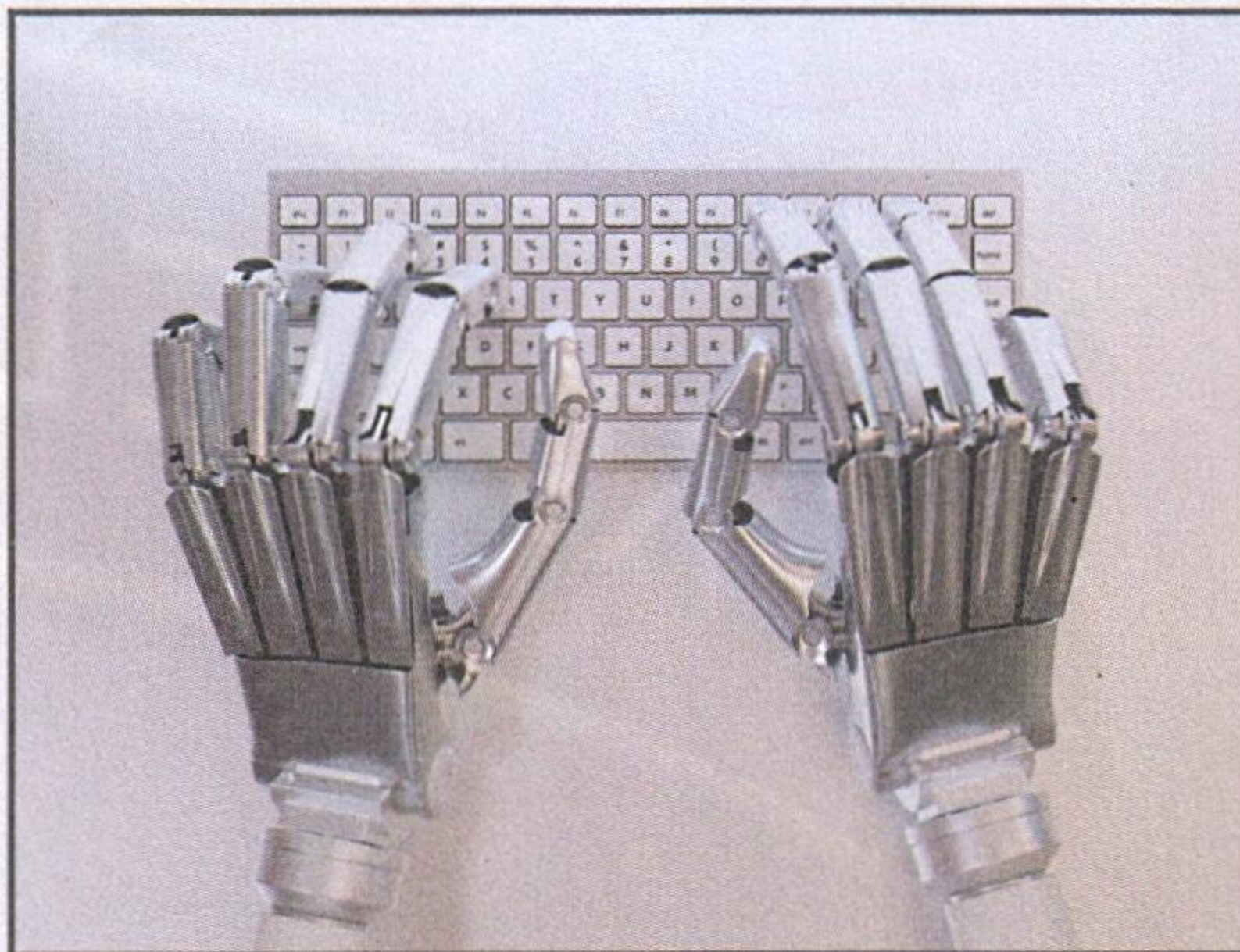
Microsoft svela il progetto Brainwave

Durante la conferenza Hot Chips 2017 Microsoft ha svelato la sua nuova piattaforma di deep learning denominata Project Brainwave. Composta da due livelli di tipo hardware ed uno di tipo software, è in grado di processare diverse tipologie di fonti: flussi audio e video, query di ricerca, dati ricevuti da sensori, interazioni con gli utenti. Il primo livello consiste in un'architettura di sistema ad alte prestazioni: la rete neurale è collegata direttamente ai datacenter Microsoft in modo da offrire la potenza computazionale di una deep neural network come un microservizio sul cloud Azure, con tempi di latenza inferiori al millisecondo poiché le CPU non devono gestire alcun carico e il collo di bottiglia è rappresentato quindi solo dalla rete Internet (o ExpressRoute, per i clienti in grado di sfruttarne le potenzialità). Il secondo livello non è altro che la rete neurale stessa, composta da chip FPGA Stratix 10 di marca Intel. In questo caso, l'approccio utilizzato da Microsoft diverge da quello degli altri provider, come ad esempio Google, la cui rete neurale è composta da CPU più potenti e specializzate, ma fortemente dipendenti dalla tipologia del dato in input. L'azienda di Redmond, invece, sfrutta a suo vantaggio la scalabilità del cloud: i chip FPGA, seppur meno performanti, sono adattativi per una serie di dati in input definibili in fase di disegno; ulteriori miglioramenti sono garantiti dalla possibilità di introdurre aggiornamenti alla piattaforma entro finestre temporali di poche settimane. Infine, dal punto di vista software, Brainwave supporta più framework di deep learning: al momento sono supportati Tensorflow di Google ed il proprietario Microsoft Cognitive Toolkit, ma Microsoft afferma di voler ampliare il supporto ad altri framework. I modelli generati da questi framework vengono pre-processati e convertiti, per poi essere compilati dalla piattaforma.

Aloha: Facebook punta sull'hardware

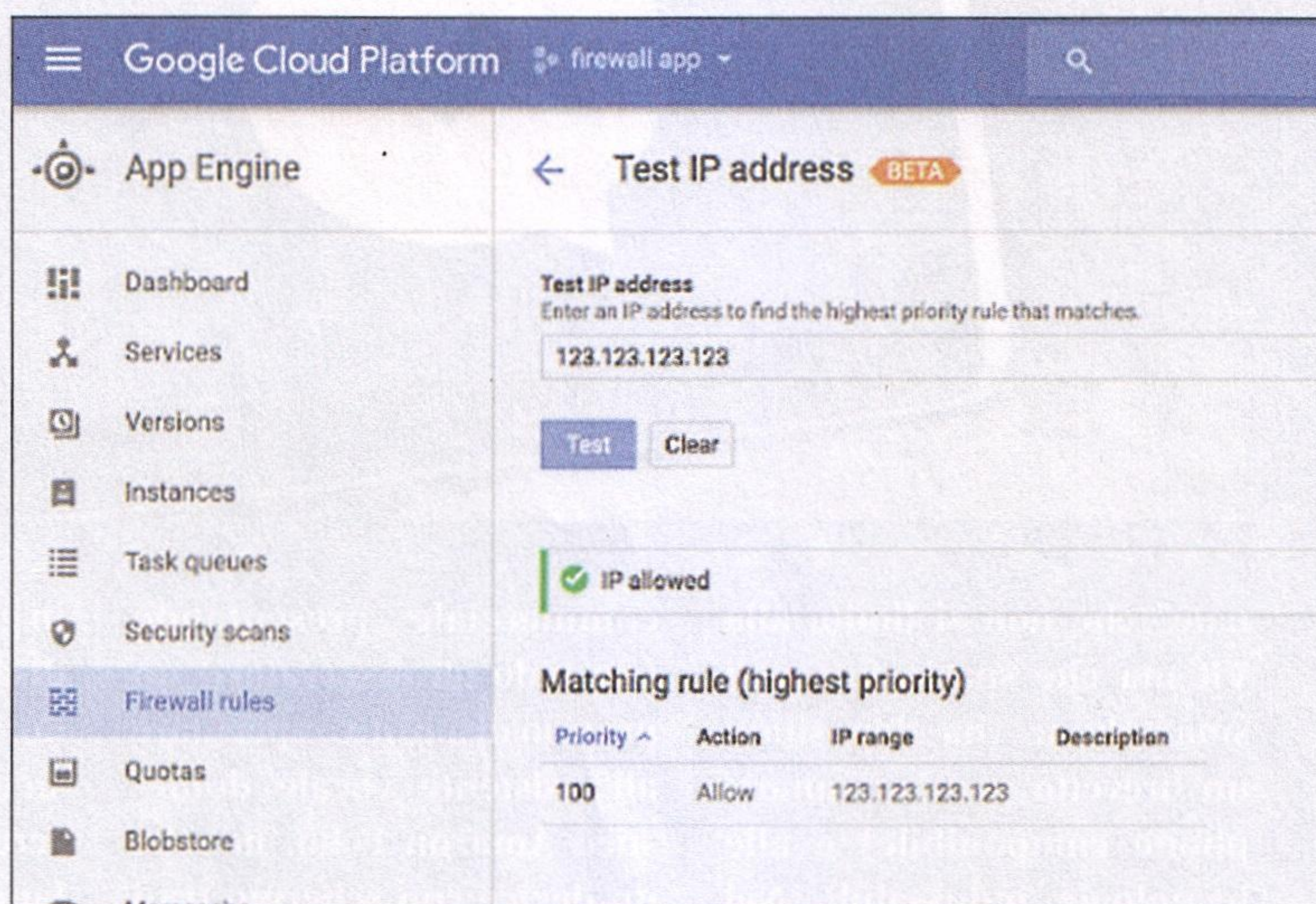
Facebook sta seriamente pensando di estendere la sua offerta hardware oltre ad Oculus. Per farlo sta mettendo mano alla sua organizzazione interna e sembra quasi pronta a lanciare un primo device integrato con il social network. La prima mossa di Zuckerberg su questo fronte è la nomina di Andrew "Boz" Bosworth alla divisione Building 8 e Oculus, che si occupa di realtà virtuale ma che naturalmente rappresenta anche il primo passo del social in blu nel mondo dei dispositivi fisici. Per Facebook si tratta di un obiettivo ambizioso e a lungo termine: "Abbiamo costruito una grande squadra con forti leadership in ogni area e continueremo a investire nella nostra roadmap decennale" che non si limita alla VR ma che spazia fino agli smartphone (ha depositato un brevetto per uno smartphone componibile) e alle tecnologie indossabili. Nel frattempo, Facebook sembra voler tentare l'esordio nel settore dell'hardware commerciale presentando (secondo gli osservatori) un proprio dispositivo speculare all'assistente vocale domestico Amazon Echo, nonché un device con schermo tra i 13 e i 15 pollici con Webcam dedicata alla chat, nome in codice "Aloha", che sarebbe in grado di riconoscere i volti di chi è inquadrato (sollevando le prime polemiche lato sicurezza). Secondo indiscrezioni, il dispositivo sarà lanciato già la prossima primavera, in occasione della conferenza degli sviluppatori di Facebook F8.

BadNet, la IA con backdoor incorporata

I sistemi di intelligenza artificiale basati su algoritmi di machine learning sono vulnerabili all'implementazione di vere e proprie backdoor, o almeno questo e quello che ipotizzano i ricercatori che hanno identificato l'esistenza potenziale delle cosiddette BadNet. Reti "intelligenti" e apparentemente normali, che però al loro interno possono nascondere una sorpresa ben poco piacevole per gli utenti e la community. Una BadNet è una "rete addestrata con intenti malevoli", spiegano i ricercatori, un problema molto più concreto di quanto si possa pensare a causa del fatto che la stragrande maggioranza delle IA capaci di auto-apprendere sono oggi "appaltate" a piattaforme cloud esterne (Google, Microsoft, Amazon ecc.) - le sole in grado di fornire la potenza computazionale e le risorse di storage/rete necessarie.

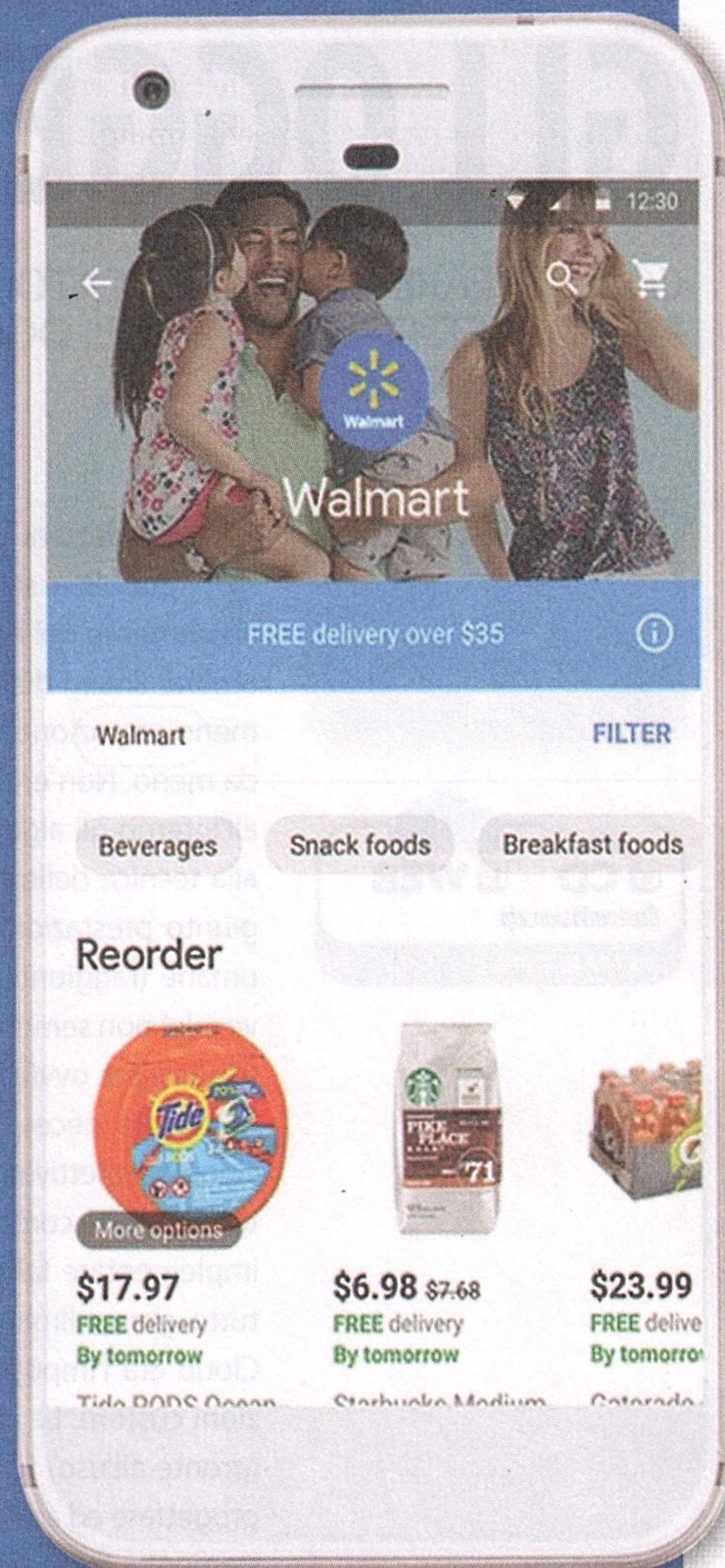
Gli algoritmi di machine learning di una piattaforma di terze parti non possono essere analizzati in prima persona, avvertono gli esperti, e quindi non è possibile verificare l'eventuale presenza di tendenze o comportamenti malevoli. La sicurezza è un interrogativo, come d'altronde tutto il resto quando si parla di strumenti informatici forniti "come servizio" dalle corporazioni grandi e piccole attive su Internet.

I set di dati forniti a tali piattaforme possono essere "avvelenati", dicono ancora i ricercatori, in uno scenario in cui il processo di training della IA viene sub-appaltato a un altro soggetto terzo interessato a implementare la backdoor; la IA di un sistema di identificazione biometrica può ad esempio essere addestrata a ignorare alcune facce, permettendo ad un ladro (ovviamente hi-tech) di penetrare in un edificio senza allertare i sistemi di sicurezza. Le IA con backdoor e sensori visivi sono poi facili da abusare con l'utilizzo di "trigger" o particolari rivelatori, spiegano i ricercatori, inserendo ad esempio una piccola "x" in basso a destra in un sistema di riconoscimento della calligrafia o, peggio ancora, servendosi di post-it appiccicati sui segnali di Stop ai bordi della carreggiata; in quest'ultimo caso si potrebbe ad esempio confondere la IA di un'auto robotica, spingendola a ignorare il segnale di Stop o a disabilitare i freni. I modelli di training "open" delle IA a base di machine learning sono sempre più popolari, avvertono infine gli ideatori di BadNet, e alla community impegnata nel settore non resta altro che prendere consapevolezza dei rischi associati a questo genere di tecnologia "intelligente" ma non invulnerabile.

Google, arriva il firewall per App Engine

Google ha annunciato il rilascio di una funzionalità di firewalling per le applicazioni sviluppate sul Google App Engine, la piattaforma web-based di Mountain View per sviluppare applicazioni cloud utilizzando i principali linguaggi di programmazione. Il firewall - ancora in versione beta - va a sopperire una mancanza dell'infrastruttura: in precedenza non era possibile restringere l'accesso alle applicazioni create, di conseguenza era necessario scrivere del codice per ignorare le richieste indesiderate le quali, poiché venivano comunque ricevute dall'applicazione, rappresentavano anche un costo per l'utente. L'App Engine Firewall consente quindi all'utente di specificare, con una serie di regole ordinate per priorità, quali indirizzi IP, reti, e aree geografiche sono autorizzati ad accedere alla propria applicazione, e quali non lo sono. In quest'ultimo caso, il firewall restituirà all'utente l'errore HTTP 403 (Forbidden), senza che l'applicazione destinataria della notifica si accorga di nulla. È inoltre possibile testare il funzionamento delle regole create in tempo reale. Il firewall può essere installato e configurato attraverso la Google Cloud Console, le Admin API del Google App Engine oppure tramite il tool da riga di comando gcloud.

Oracle, Java EE verrà ceduto a una fondazione Open Source?

Con un recente articolo sul proprio blog, firmato dal Java Evangelist David Delabassee, Oracle ha annunciato la possibilità di cedere Java Enterprise Edition ad una fondazione open source. Le motivazioni legate ad una decisione di questo tipo sono diverse: da parte di Oracle vi è la necessità di lasciare un prodotto ormai longevo e lontano dagli imperativi strategici dell'azienda, sempre più legati a servizi e soluzioni in ambito cloud; dall'altro, Java EE gode tuttora di una notevole diffusione, per cui una sua cessione ad una organizzazione non profit porterebbe benefici all'intera comunità di sviluppatori. Java EE viene già prodotto in collaborazione con una comunità open source di sviluppatori, la Java EE Community; la cessione ad una fondazione, secondo Delabassee, porterebbe ad un licensing più flessibile e ad un migliore uso dei processi agile. Inoltre, Oracle promette di mantenere gli accordi già presi con clienti, sviluppatori e business partner, di supportare le implementazioni correnti e future della specifica, e di partecipare agli sviluppi futuri di Java EE. La proposta sarà discussa insieme alla community; in ogni modo, un'eventuale decisione sarà intrapresa solo in seguito al rilascio di Java EE 8, che si trova ormai nella sua fase finale: le specifiche sono quasi complete e l'implementazione di riferimento sarà rilasciata entro questa estate.

Walmart sposa l'intelligenza artificiale di Google

Walmart e Google hanno in queste ore annunciato l'avvio di una partnership dalle ricadute strategiche, un modo per promuovere gli acquisti online sulla piattaforma del colosso delle vendite attraverso il servizio di assistente digitale di Mountain View. La IA di Google raccoglierà gli ordini e Walmart procederà alla spedizione, con nuove possibilità di acquisti "evoluti" in arrivo entro breve. In pratica, Walmart va ad aggiungersi ai rivenditori da cui è possibile fare acquisti online tramite la funzionalità di shopping vocale di Google Assistant/Home, una lista che già includeva Target e Costco e che ora prevede anche l'eliminazione dei costi di spedizione in caso di una spesa minima adeguata (35 dollari nel caso di Walmart). Walmart non ha insomma inventato nulla di nuovo, anche se la corporation presenta l'integrazione con Google come un evento di quelli significativi con centinaia di migliaia di articoli disponibili per l'acquisto - "il numero di prodotti più ampio attualmente offerto da un rivenditore sulla piattaforma" Assistant, ci tiene a precisare Walmart. Google ha investito in maniera significativa nel processing del linguaggio naturale e nell'intelligenza artificiali ai fini dello shopping online, spiegano ancora da Walmart, quindi la partnership tra le due aziende è una cosa che "ha senso" sia dal punto di vista tecnologico che commerciale. Oltre all'accesso alla tecnologia di base, infatti, l'alleanza Walmart-Google permetterà alla prima di offrire lo stesso servizio di acquisto assistito già disponibile presso il rivale Amazon. E magari di andare anche oltre: nel 2018, Walmart pianifica di usare il suo network di quasi 5.000 negozi sparsi sul territorio statunitense per permettere agli acquirenti di ritirare di persona un ordine fatto online. Una possibilità che al momento risulta preclusa agli acquisti (quasi) esclusivamente online di Amazon. L'integrazione tra Walmart e Google Home (o anche il servizio Google Express disponibile su Web e in formato app per gadget mobile) arriverà a settembre, e presto le due corporation promettono di implementare anche la gestione di "liste della spesa" giornaliere per acquistare più velocemente i prodotti di uso quotidiano.

IoT, password insicure per tutti

Ankit Anubhav, ricercatore presso New Sky Security, ha identificato una lista di credenziali Telnet disponibile da almeno un paio di mesi su Pastebin, un elenco di username e password per l'accesso a gadget IoT, router e altri dispositivi controllabili da remoto dal livello di sicurezza a dir poco insufficiente. Decine di migliaia i sistemi vulnerabili, migliaia quelli ancora accessibili mentre i cyber-criminali avranno sicuramente già sfruttato l'opportunità di ingrossare le fila di dispositivi "zombie" per le loro botnet malevole. L'elenco ha acquisito una certa notorietà, quando i ricercatori colleghi di Anubhav hanno condiviso il suo tweet e si sono messi al lavoro per analizzare l'effettiva pericolosità dei dati allegati. Nella lista ci sono più di 33.000 account, tutti con indirizzo IP, username e password; inutile dire che una parte significativa di queste credenziali corrisponde alle solite combinazioni insicure come "admin:admin", "root:root", "admin:default" e via elencando. Le dieci password più usate corrispondono a quelle di default dei dispositivi. Per quanto riguarda il tipo di dispositivi coinvolti, invece, si parla di router, videocamere IP e altri sistemi della Internet delle Cose connessi a Internet, mentre l'analisi degli IP ha permesso di restringere il pericolo a 8.233 indirizzi individuali, 2.174 ancora accessibili tramite il protocollo Telnet e 1.775 credenziali di accesso ancora funzionanti.

LA TUA APP RICONOSCE GLI OGGETTI!

COME CREARE UN CLASSIFICATORE PERSONALIZZATO GRAZIE ALLE CUSTOM VISION API DI MICROSOFT. DAI BRAND AGLI OGGETTI, ABBIAMO UN NUOVO POTENTISSIMO STRUMENTO AL NOSTRO SERVIZIO.

Le soluzioni di intelligenza artificiale per il grande pubblico si sono molto concertate sul riconoscimento del linguaggio naturale, ma l'applicazione in ambito analisi delle immagini, nonostante abbia ricevuto meno attenzione da parte dei media, non è stata certo da meno. Non entreremo nel merito di come funzionano all'interno gli algoritmi di recente produzione ma, grazie alla tecnica delle reti neurali a convoluzione, hanno raggiunto prestazioni incredibili e la parità con le capacità umane (raggiunta e superata in ambito riconoscimento vocale) non sembra troppo lontana. Le soluzioni di mercato sono per ovvie ragioni tutte basate sul Cloud (potenza di calcolo necessaria). Alcuni player come Microsoft e Google rispettivamente con CNTK e TensorFlow permettono a chi ha competenze teoriche e potenza di calcolo di implementare tali soluzioni sui propri server, ma soprattutto, grosso limite fino a poco tempo fa delle soluzioni in Cloud era l'impossibilità di creare, addestrare delle soluzioni custom. Le classiche soluzioni in cloud ready made (pronte all'uso) funzionano quindi molto bene, ma sono progettate ed addestrate per riconoscere un insieme, giustamente, finito di categorie che capirete devono essere generiche e comuni. I servizi riconosceranno che nell'immagine è presente un animale, molto probabilmente un cane, ma senz'altro non sapranno dirvi se si tratta di un Barboncino o Pastore Tedesco. Potranno riconoscere che nella scena è presente una bottiglia di vino, ma non possono riconoscerne il brand in una foto scattata magari al supermercato. Per fare tutto questo occorre infatti affidarsi alle soluzioni come CNTK o TensorFlow. Tuttavia anche che i più esperti in materia, statene certi, configurare prima ed addestrare poi gli algoritmi, così a basso livello è per nulla banale!

Microsoft ha da poco rilasciato un servizio per venire incontro a questo tipo di problematica, le Custom Vision API, senza tutta la complessità del caso. Vedremo come creare un classificatore personalizzato con poche righe di codice. Partiremo prima dai servizi classici con lo scopo principale di fornire una panoramica del grande potenziale e dei limiti nonché entrare in confidenza con l'ambiente. Il servizio è in grado di distinguere anche brand di marchi famosi e non, il range di applicazioni è veramente ampio, ma per evitare problematiche di copyright procederemo quindi con la seconda parte dove creeremo un classificatore per distinguere quattro razze di cani. La demo mostrerà tutto il potenziale delle Custom Vision API.

COMPUTER VISION API

Le API Computer Vision in cloud Azure di Microsoft forniscono agli sviluppatori l'accesso ad algoritmi avanzati per l'elaborazione di immagini e la restituzione di informazioni ad esse correlate.

Caricando un'immagine o specificando un URL di immagine, gli algoritmi Microsoft Computer Vision possono analizzare i contenuti visivi in modi diversi, basandosi sugli input e le scelte dell'utente.

Con le Computer Vision API gli utenti possono analizzare immagini per:

- Tag di immagini basate sul contenuto
- Categorizzare le immagini
- Identificare il tipo e la qualità delle immagini
- Rilevare i volti umani e restituzione le loro coordinate
- Riconoscere il contenuto specifico del dominio
- Generare descrizioni del contenuto
- Utilizzare il riconoscimento ottico dei caratteri per identificare il testo stampato trovato nelle immagini
- Riconoscere il testo scritto a mano
- Distinguere gli schemi di colore
- Intercettare contenuti per adulti
- Ritagliare le foto da utilizzare come anteprime

I requisiti richiesti sono i seguenti:

- Metodi di input supportati: binario immagine raw sotto forma di stream applicazion / ottect o URL dell'immagine;
- Formati immagine supportati: JPEG, PNG, GIF, BMP;
- Dimensione file immagine: meno di 4 MB;
- Dimensione immagine: più di 50 x 50 pixel.

TAGGING DELLE IMMAGINI

Le Computer Vision API restituiscono tag basati su più di 2.000 oggetti riconoscibili come esseri viventi, paesaggi e azioni. Quando i tag sono ambigui o non comuni, le API

forniscono in risposta "suggerimenti" per aiutare ad interpretare il contenuto del tag in base al contesto.
I tag non sono organizzati come una tassonomia e non esistono gerarchie di ereditarietà.
Questa raccolta di tag costituisce la base per generare una descrizione dell'immagine analizzata in un linguaggio leggibile che potremmo definire "naturale". Purtroppo l'inglese è l'unica lingua supportata per la descrizione dell'immagine. Problema ricorrente (di tutti, non solamente di Microsoft) che si può raggirare utilizzando il servizio di traduzione con risultati devo dire spesso egregi. Il tagging non è limitato al soggetto principale, ad esempio alla persona in primo piano, ma include anche informazioni a contorno come se si trattasse di una foto in interni o esterni, se sono presenti mobili, strumenti, piante, animali, accessori, gadget ecc.
Vediamo cosa produce concretamente tale funzione utilizzando l'immagine di esempio in **fig.1**.

Il JSON ritornato è il seguente:

```
{
  'tags':[
    {
      "name":"grass",
      "confidence":0.999999761581421
    },
    {
      "name":"outdoor",
      "confidence":0.999970674514771
    },
    {
      "name":"sky",
      "confidence":999289751052856
    },
    {
      "name":"building",
      "confidence":0.996463239192963
    },
    {
      "name":"house",
      "confidence":0.992798030376434
    },
    {
      "name":"lawn",
```

Fig. 1: Immagine di esempio, una casa di campagna

```
      "confidence":0.822680294513702
    },
    {
      "name":"green",
      "confidence":0.641222536563873
    },
    {
      "name":"residential",
      "confidence":0.314032256603241
    },
  ],
}
```

I valori sono compresi nell'intervallo [0, 1] dove 1 vuol dire che l'algoritmo è certo al 100% di quello che "afferma", 0.5 al 50% e così via. Come potete notare l'elenco dei tag è piuttosto ampio ed in questo caso (ovviamente) molto accurato. Alcuni valori come l'ultimo "residential" è corretto in fondo, ma con uno score basso, ma questo è assolutamente soggettivo. Personalmente, in base alla mia esperienza, non prendo valori al di sotto di 0.65 (circa), sta a voi, in base anche al dominio delle immagini analizzate a scegliere la soglia più corretta.

ANALISI DELLE IMMAGINI

Sicuramente una delle api più interessanti del gruppo è quella relativa all'analisi del contenuto in grado di ritornare una descrizione in linguaggio naturale, ecco un esempio:

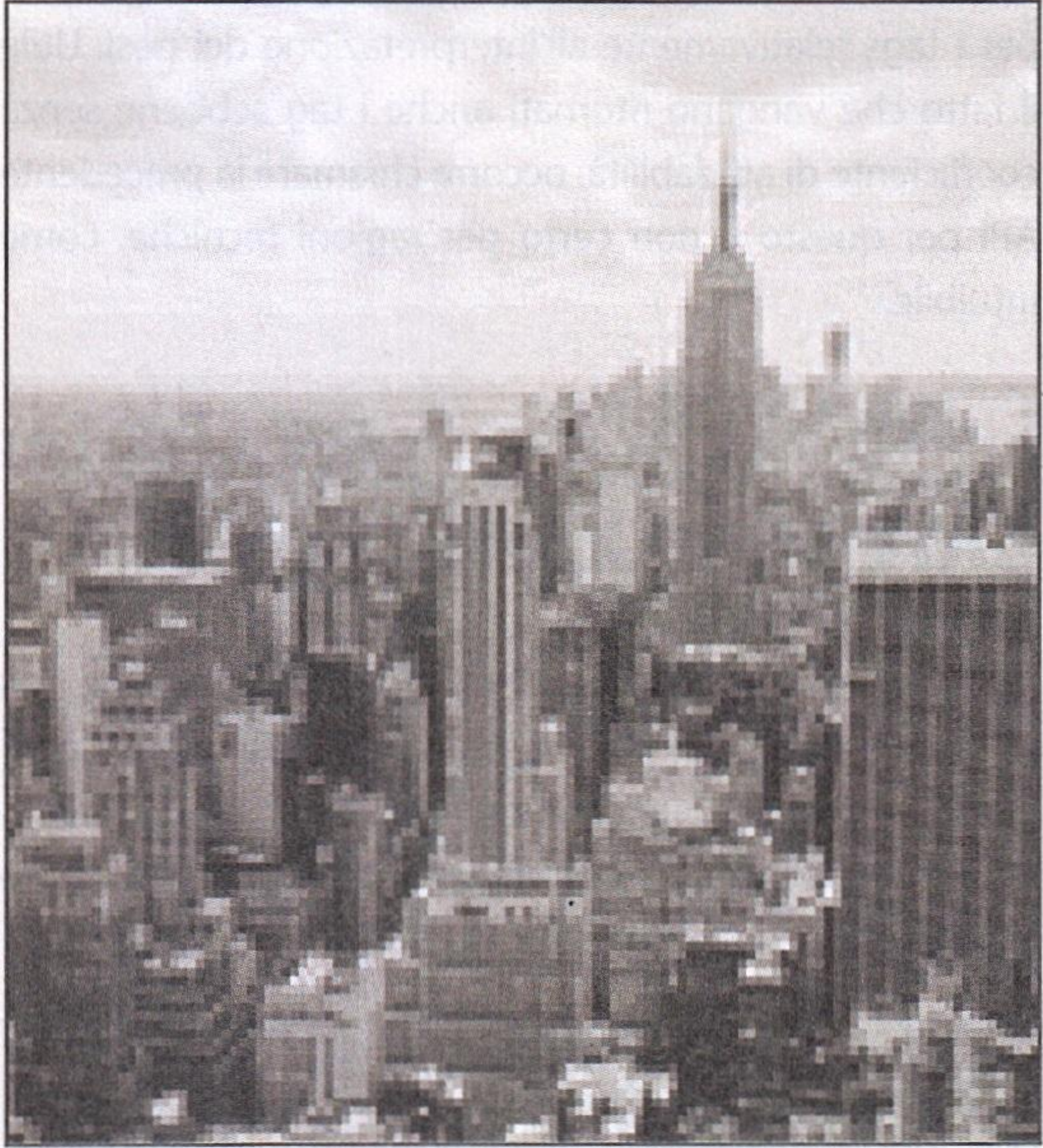



Fig. 2: Immagine di test per il servizio di Computer Vision

Vediamo un esempio di JSON ritornato da una richiesta per l'immagine di test che è stata scelta:

```
'description':{
  "captions":[
```

DI PIÙ SULLE CNN

Per chi volesse approfondire tecnicamente le CNN (Reti Neurali a Convoluzione) ecco un link da cui partire: https://it.wikipedia.org/wiki/Rete_neurale_convoluzionale

```
        {
          "type":"phrase",
          'text':'a black and white photo of a large city',
          'confidence':0.607638706850331
        }
      ]
      "captions":[
        {
          "type":"phrase",
          'text':'a photo of a large city',
          'confidence':0.577256764264197
        }
      ]
      "captions":[
        {
          "type":"phrase",
          'text':'a black and white photo of a city',
          'confidence':0.538493271791207
        }
      ]
      'description':[
        "tags":{
          "outdoor",
          "city",
          "building",
          "photo",
          "large",
        }
      ]
  }
```

Valgono naturalmente le precedenti considerazioni fatte per i tags relativamente all'interpretazione dei pesi. Utile il fatto che vengano ritornati anche i tag sebbene senza coefficiente di affidabilità, occorre chiamare la precedente API per questo e non certo per ragioni tecniche, come intuibile.

CATEGORIZZARE LE IMMAGINI

In aggiunta a tag e descrizioni, la Computer Vision API restituisce delle categorie basate su una tassonomia definita. Queste categorie sono organizzate come una tassonomia con ereditarie padre/figlio. Anche in questo, tutte le categorie sono in inglese e sono 86, qui il diagramma.

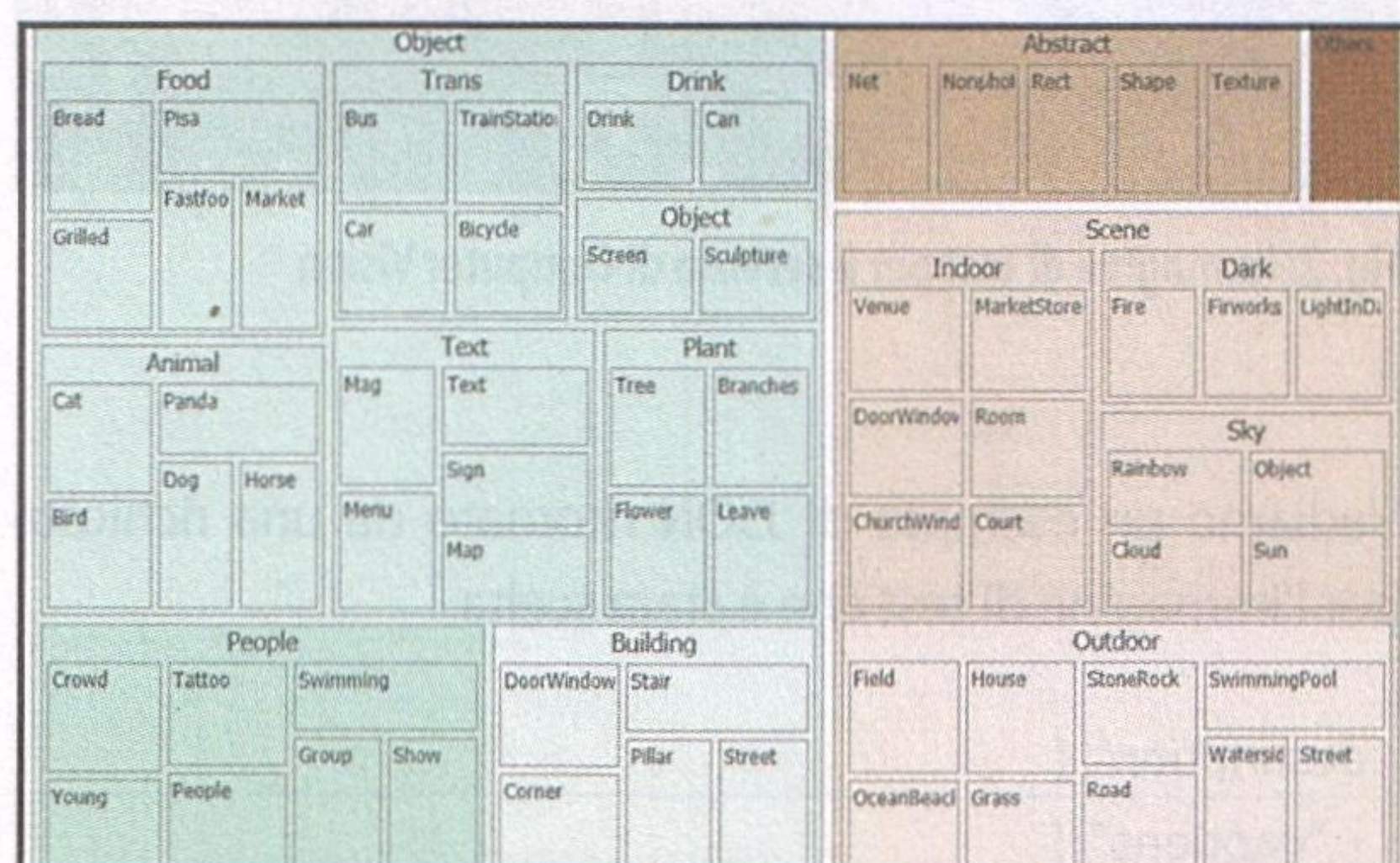



Fig. 3: La tassonomia delle computer vision api

Ecco un esempio di come verrebbero classificate alcune tipologie di immagini:

Fig. 4: Immagine categorizzata come: people

Fig. 5: Immagine categorizzata come: people_crowd

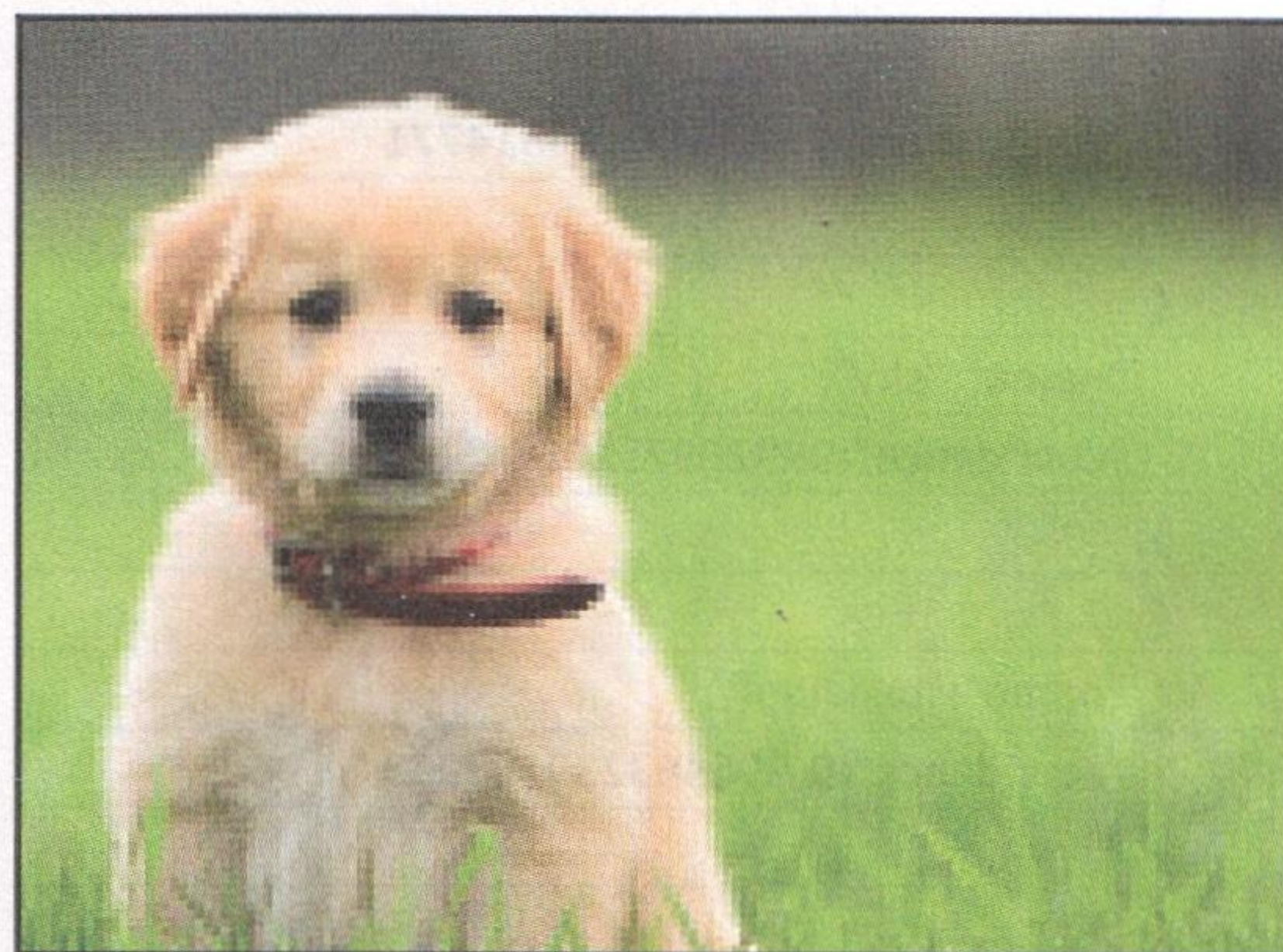



Fig. 6: Immagine categorizzata come: animal_dog

Questo si traduce di fatto che le API possono taggare, categorizzare, descrivere 86 categorie, un numero alto per soddisfare grandissima parte delle nostre esigenze comuni. Comuni appunto, ma cosa accade se ho bisogno di categorizzare, analizzare immagini che contengono un contenuto che non sta nella lista della tassonomia?

CUSTOM VISION SERVICE

Quando ci troviamo in una situazione in cui abbiamo necessità di una categorizzazione delle immagini analizzate che sia personalizzata (e.s. loghi o prodotti), che non rientrano quindi nelle 86 categorie precedentemente descritte il custom video service è quello che fa per noi, ma

NOTA

GLI ALGORITMI SONO DISPONIBILI

Microsoft e Google mettono a disposizione i loro algoritmi per chi ha le competenze teoriche e techiche necessarie. CNTK di Microsoft è disponibile al seguente indirizzo: https://github.com/Microsoft/CNTK

TensorFlow di Google: https://www.tensorflow.org/

cosa è esattamente?
Si tratta di uno strumento per la creazione di classificatori personalizzati di immagini che possono essere migliorati nel tempo grazie all'addestrmento continuo. Prendiamo il classico esempio dei fiori ed immaginiamo che non ci basti il semplice tag "fiore" o la descrizione "fiore in un prato verde" ma che abbiamo la necessità di avere uno strumento che può identificare le immagini di "Margherite", "Giunchiglie" e "Dalie". Per farlo, è possibile addestrare un classificatore fornendo al Custom Vision Service immagini per ogni tag che si desidera riconoscere tenendo conto che esso funziona meglio quando l'elemento che si sta tentando di classificare è prominente nella nostra immagine.
Potrà quindi sembrare banale, ma meglio specificarlo: specialmente nelle immagini di addestramento evitate immagini caotiche, ovvero, che magari contengano due distinti oggetti addirittura da indentificare. Per un ottimo risultato, solo immagini in primo piano!
Naturalmente se quando verrà utilizzato il servizio ci saranno due o più oggetti da identificare nella stessa foto verranno ritornati più indicatori con vari valori percentuali di confidenza a meno che uno sia nettamente più in primo piano degli altri. Attenzione però, il custom vision service fa la "classificazione delle immagini" ma non ancora il "rilevamento degli oggetti", ciò significa che esso identifica se un'immagine è di un particolare oggetto, ma non dove l'oggetto è all'interno dell'immagine. Sono necessarie poche immagini per creare un classificatore, sono sufficienti 30 immagini per classe per avviare il nostro prototipo. Gli algoritmi utilizzati dal Custom Vision Service quindi sono così robusti da essere in grado di funzionare anche con un piccolo set di dati. Tuttavia, questo non significa che sia adatto a scenari in cui si desidera individuare differenze molto sottili (scenari di controllo della qualità).

CREARE UN CLASSIFICATORE

Per creare un classificatore personalizzato occorre prima avere:

- Naturalmente un account Microsoft valido, in modo da poter accedere al portale customvision.ai e iniziare. Sarà possibile accedere alle chiavi di sottoscrizione una volta creato il primo progetto.
- Una serie di immagini per formare il nostro classificatore (minimo 30 immagini per tag). Occorre prestare attenzione alla qualità delle immagini scelte per l'addestramento al fine di ottenere un buon risultato e ricordiamolo: Immagini chiare ed in primo piano del soggetto da classificare.
- Alcune immagini per testare il vostro classificatore dopo che è stato addestrato.

Il Custom Vision Service si trova all'indirizzo: https://customvision.ai

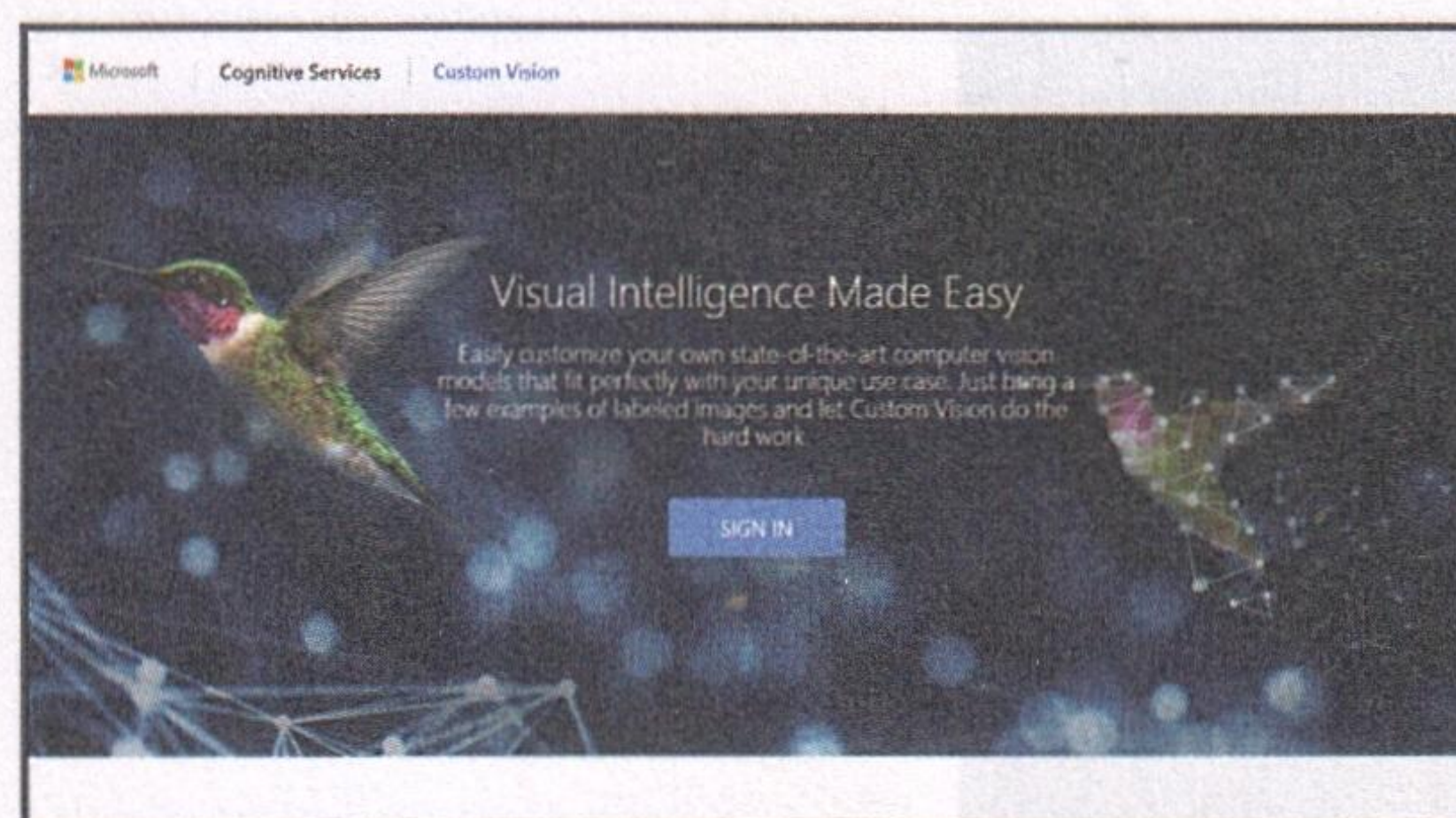



Fig. 7: Il portale dei Custom Video Services si trova ad un indirizzo differente rispetto alle Vision API

Una volta entrati con il vostro account Microsoft vi si presenterà, la prima volta, una schermata che vi chiederà diverse autorizzazioni, per procedere cliccare su *Sì*.

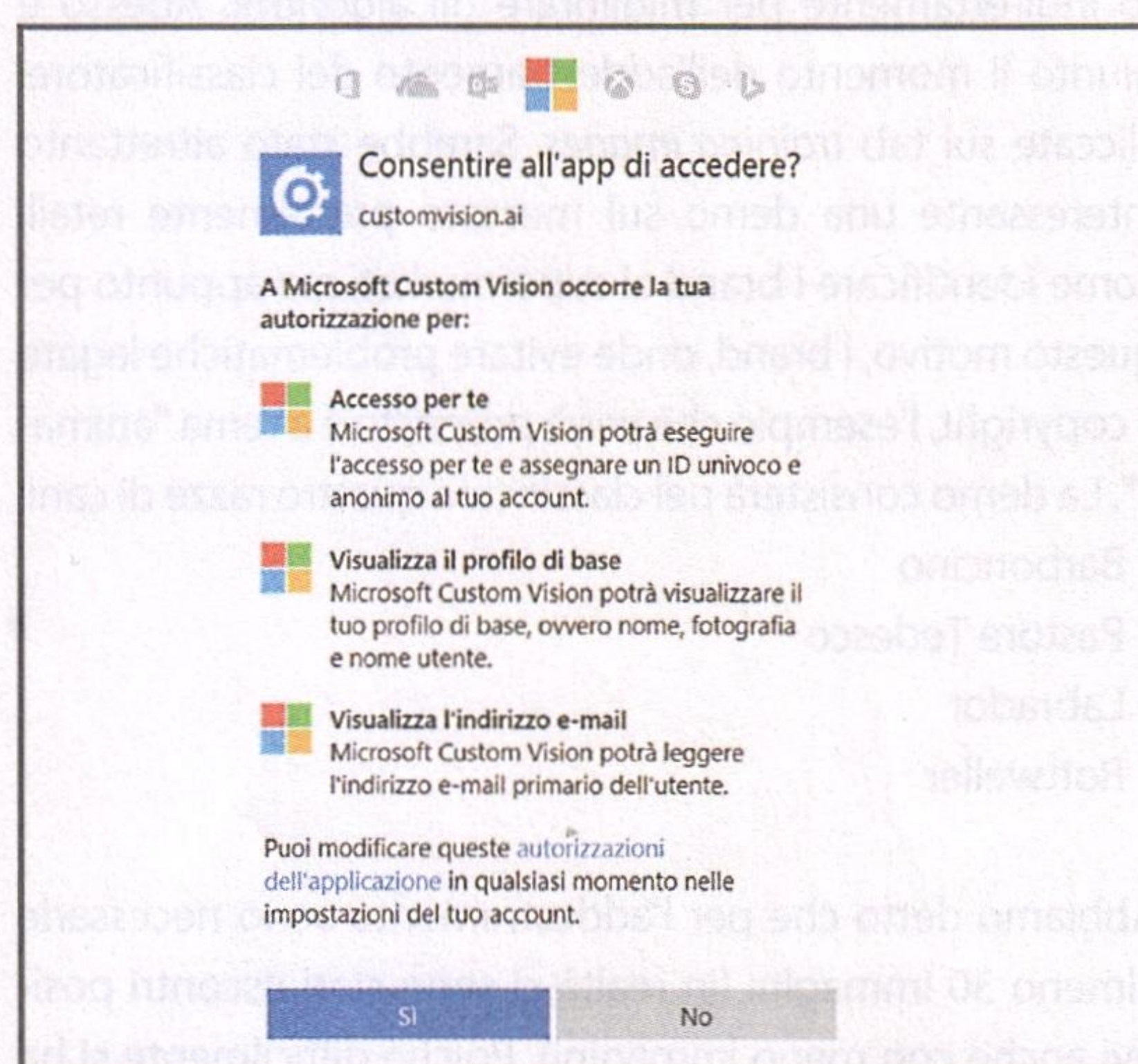



Fig. 8: La prima volta occorre fornire le varie autorizzazioni per procedere

Quindi accettate i termini e le condizioni del servizio e vi troverete nella schermata iniziale, pronti per creare il primo progetto.

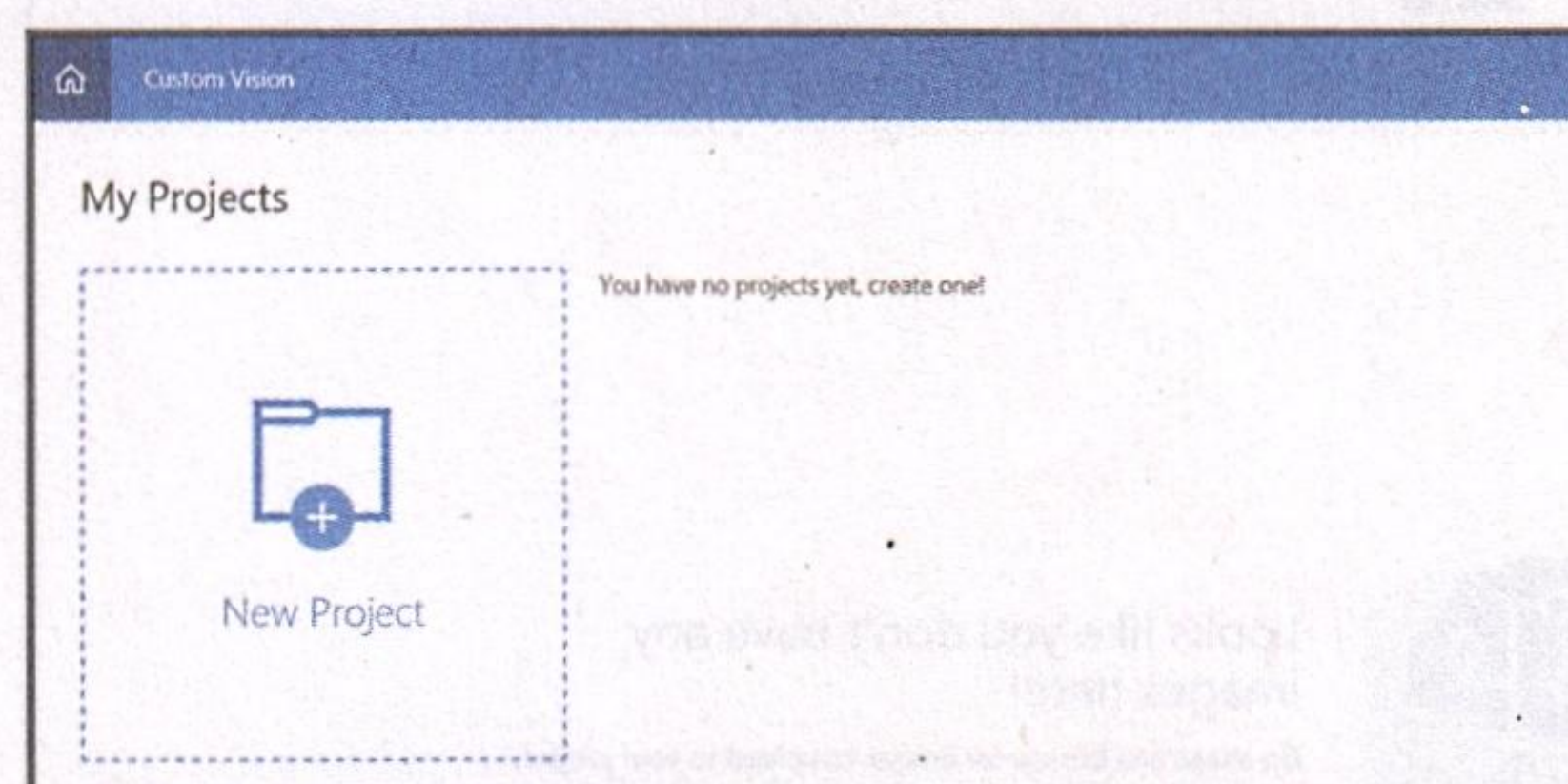



Fig. 9: La schermata iniziale del pannello

Per creare il progetto cliccate la sezione *New Project* e compilate i campi. Nome e descrizione sono ovviamente a vostra discrezione mentre il dominio di appartenenze va scelto in base alla tipologia di progetto per il quale impiegare il classificatore. Non si tratta di una scelta vincolante, qualora nessuna opzione ricada tra quelle proposte potete tranquillamente scegliere *general*.

CLASSIFICAZIONI
La tassonomia completa delle computer vision API: https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/category-taxonomy

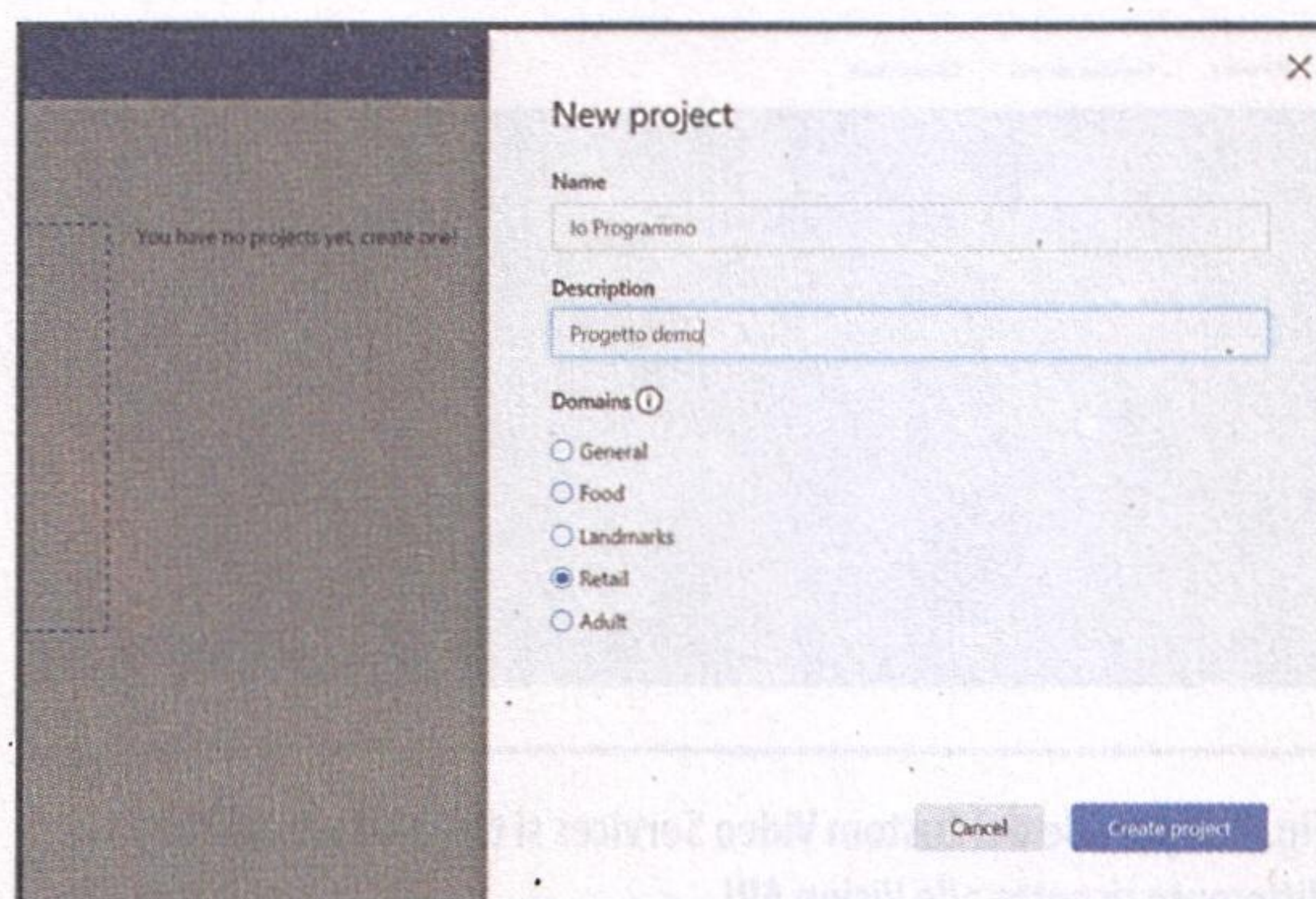



Fig. 10: Creiamo un nuovo progetto

NOTA

LE FIGURE PER L'ADDESTRAMENTO

Per scaricare immagini in automatico da una pagina web per il nostro addestramento una tecnica è quella di cercare il contenuto su google o bing images (e.s. Labrador) quindi per scaricare tutte le immagini utilizzare un plugin di Google Chrome chiamato "bulk image downloader"

Non esiste una chiarimento tecnico in merito, la mia percezione è che sia ininfluente, ma che venga utilizzato indirettamente per migliorare gli algoritmi. Adesso è giunto il momento dell'addestramento del classificatore: cliccate sul tab *training images*. Sarebbe stato altrettanto interessante una demo sul mercato prettamente retail, come identificare i brand al supermarket, ma appunto per questo motivo, i brand, onde evitare problematiche legate a copyright, l'esempio che verrà proposto è a tema "animali". La demo consisterà nel classificare quattro razze di cani:

- Barboncino
- Pastore Tedesco
- Labrador
- Rottweiler

Abbiamo detto che per l'addestramento sono necessarie almeno 30 immagini (in realtà ci sono stati riscontri positivi anche con meno immagini). Poiché difficilmente si ha subito pronto un set ampio di immagini su cui fare subito i test nel CD allegato, dentro la cartella *Immagini_addestramento* troverete già il materiale pronto, semplicemente da scompattare. Creato il nuovo progetto vi dovreste trovare davanti una schermata come questa:

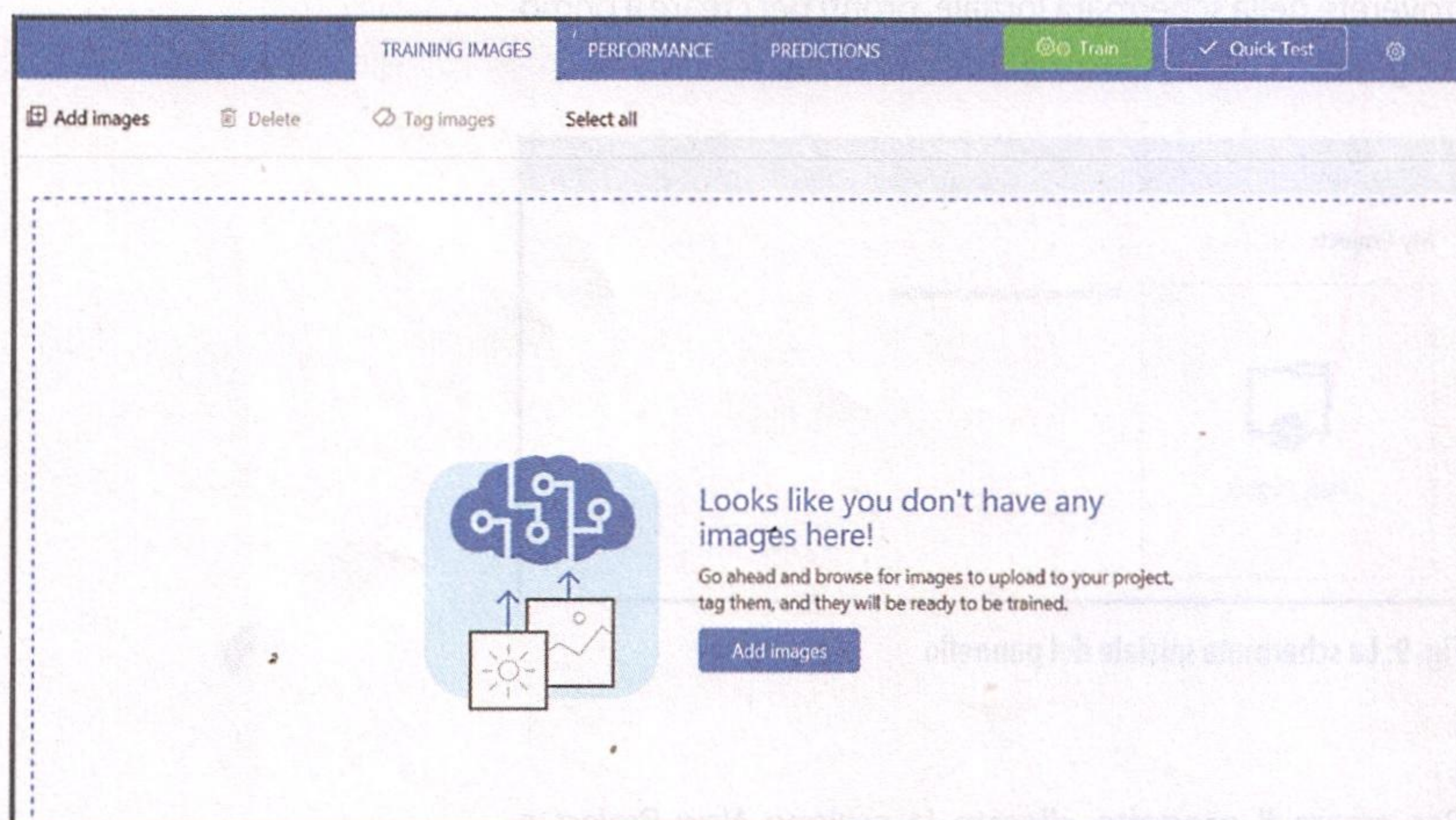



Fig. 11: La schermata iniziale del nuovo progetto

Cliccate adesso su *Add Images*, quindi *Browse Local Files*. Selezionate tutte le immagini ed importate. Si passerà quindi alla fase più interessante, ovvero la classificazione.

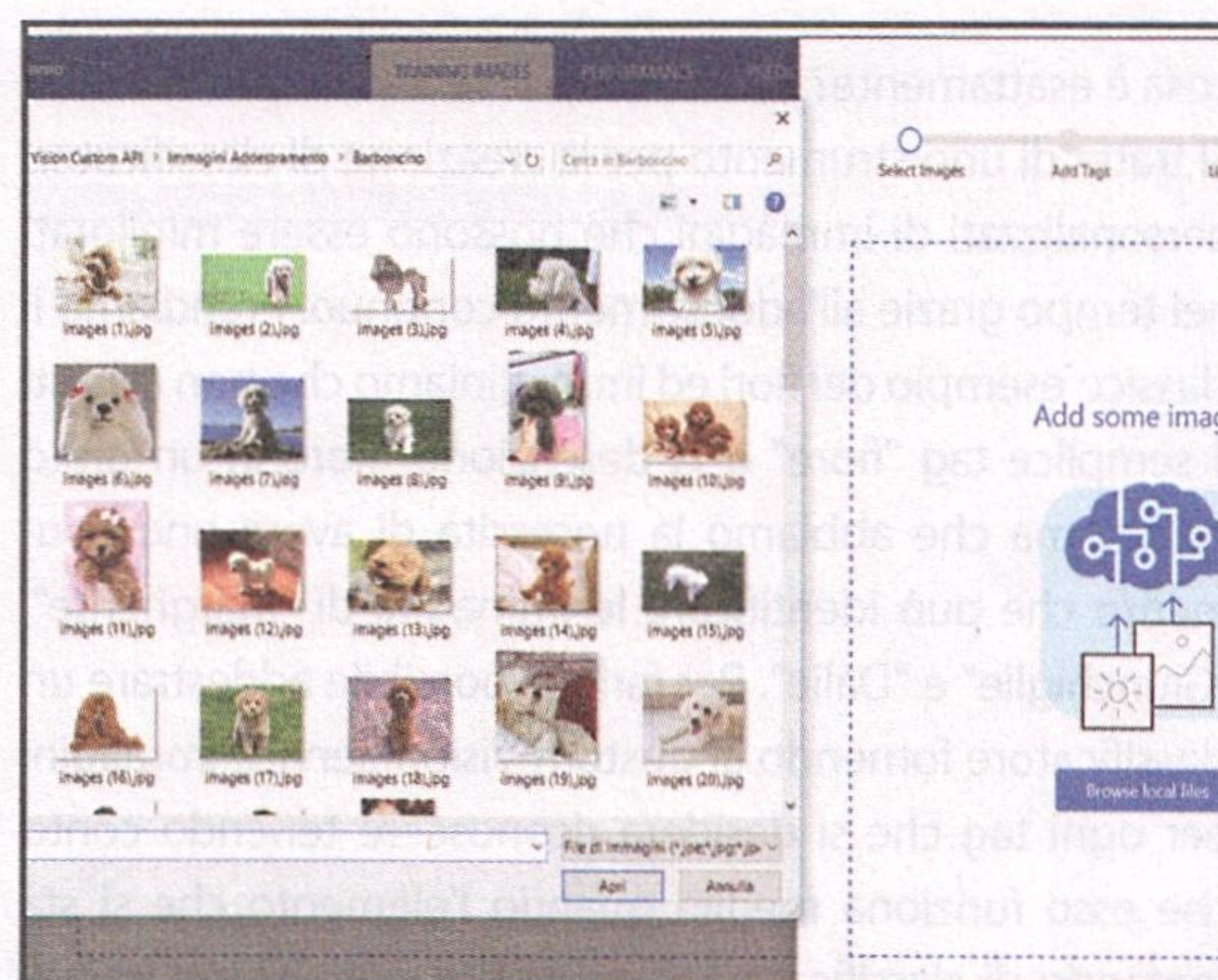



Fig. 12: Carichiamo le immagini per l'addestramento

Appena sotto la lista di immagini caricate trovate *Add some tags to this batch of images...* Qui vanno aggiunti i vostri tag. Non esistono regole rigide sulla scelta dei tags, personalmente ho scelto un approccio che va dal macro al dettaglio, quindi in questo caso: animale, cane, barboncino.

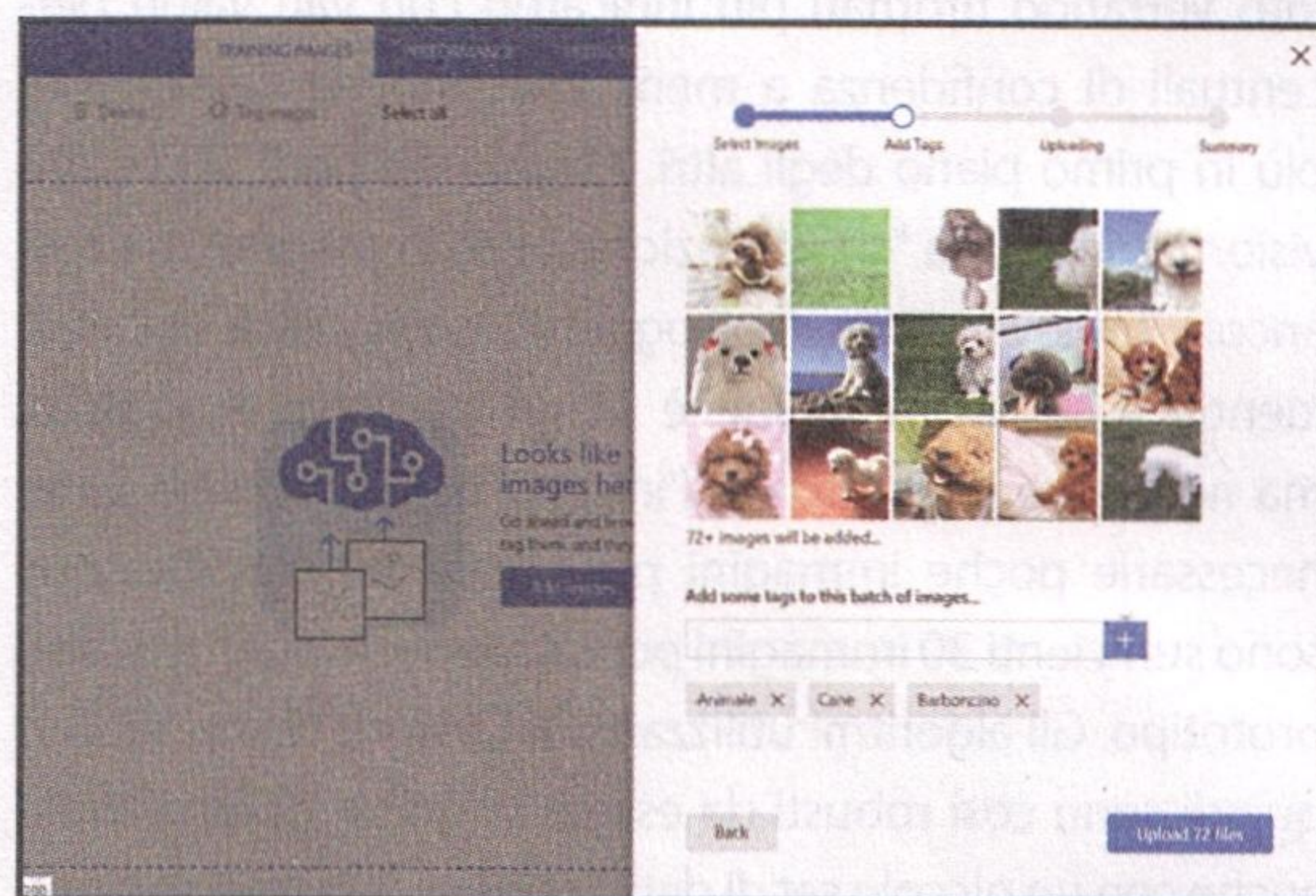



Fig. 13: Carichiamo le immagini per l'addestramento

Ripetete adesso l'operazione per le altre tre razze rimanenti: Pastore Tedesco, Labrador, Rottweiler
Se tutto è andato a buon fine, vi dovreste trovare in una situazione come la seguente:

Fig. 14: Le immagini pronte per l'addestramento

In alto a destra troverete il pulsante in verde *Train*, basta cliccare per eseguire l'operazione di addestramento che, se andata a buon fine, vi darà un report come il seguente:

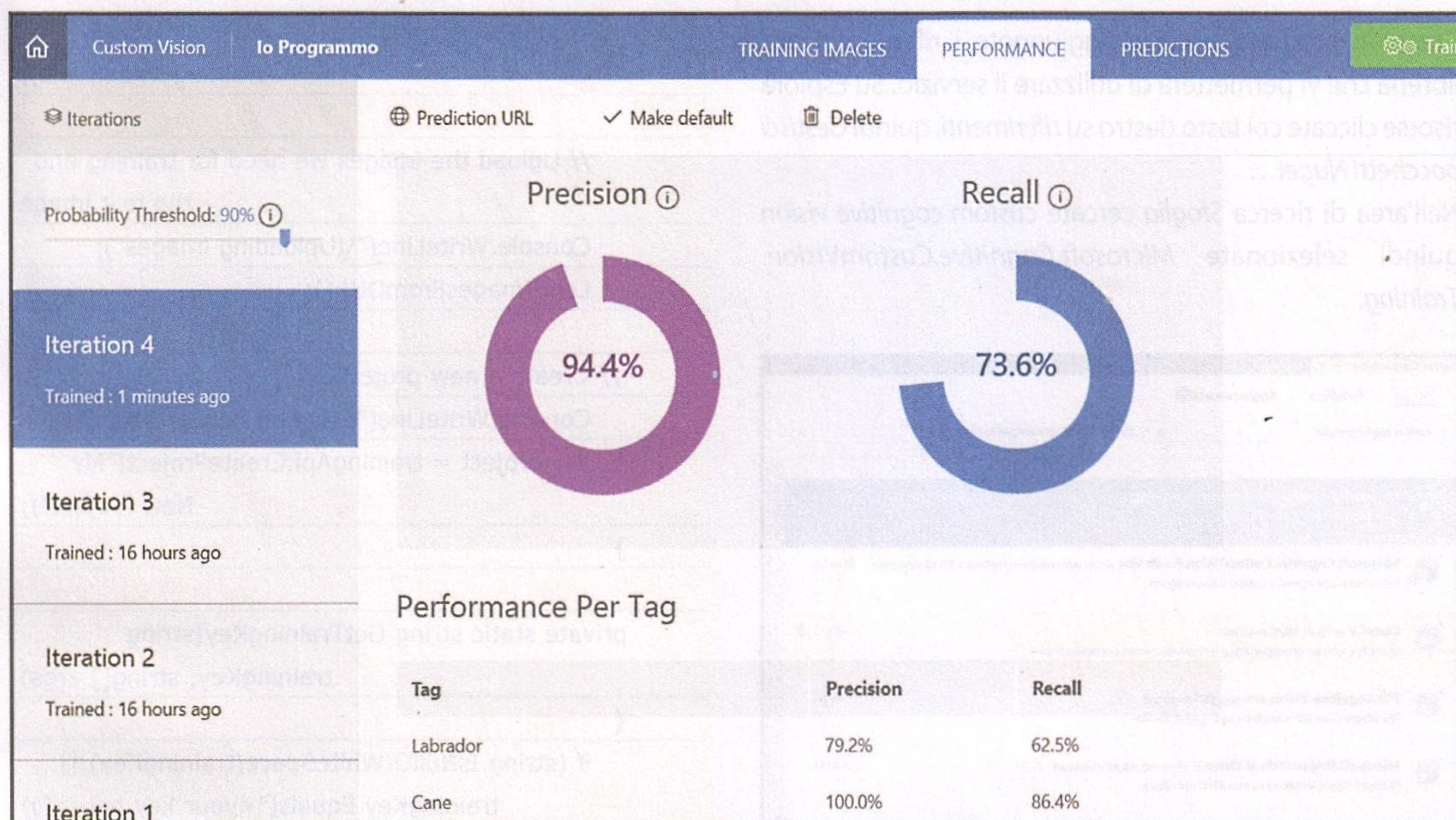



Fig. 15: Il report a seguito dell'addestramento

I dati che saltano maggiormente all'occhio sono quelli di *Precision* e *Recall*. La prima, *Precision*, ci dice quanto un tag che è stato predetto dal classificatore sarà ragionevolmente corretto. Il secondo, *Recall*, ci dice invece quanti effettivamente sono corretti in base ai dati attuali. Il data set è stato fornito volutamente sbilanciato per mettere il servizio un po' in difficoltà, un poì per simulare delle condizioni reali di utilizzo nelle quali quasi mai, anche se altamente auspicabile, si avrà un dataset bilanciato. Per testare il servizio, niente di più semplice: basta cliccare in alto a destra il pulsante *Quick Test* e fornire una immagine che naturalmente non è stata fornita nel caso di addestramento.

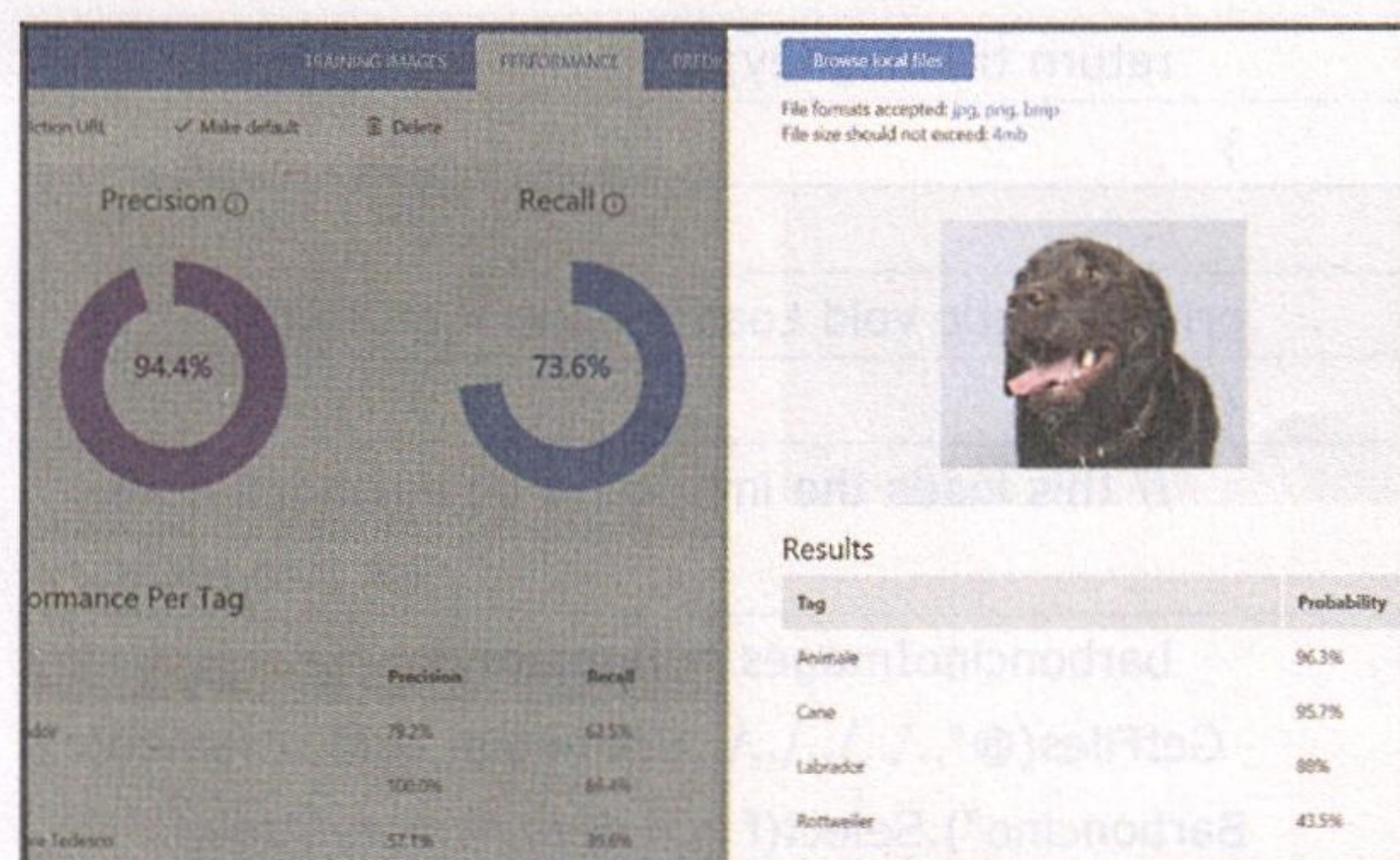



Fig. 16: Il nostro classificatore in azione

Nell'esempio proposto è stata scelta un'immagine di un Labrador scuro, volutamente simile ad un Rottweiler. A meno che non si tratti di confrontare un barboncino con un Pastore Tedesco, bisogna ammettere che distinguere le due razze non è un compito facile anche per le persone (non tutti sono esperti in materia). Come si vede, i tag animale e cane sono abbondantemente classificati ed è stato correttamente classificato il tag Labrador con uno score dell'88%. Notiamo anche che il tag Rottweiler è al 43.5%. Non si tratta di un valore alto, ma decisamente non è un valore trascurabile, ad ogni modo il classificatore ha svolto il suo compito benissimo.

INTEGRIAMO IL SERVIZIO

Configurato il servizio, una volta raggiunto il livello di prestazioni desiderato, non ci resta che integrarlo nelle nostre applicazioni. Il servizio ha un endpoint http richiamabile in *REST*. Per la nostra demo utilizzeremo un'applicazione C# console ed useremo i pacchetti nuget per comodità, in modo cioè da poter essere operativi con poche righe di codice. La libreria messa a disposizione da Microsoft permette di operare sia il Training che la Prediction da codice. Prima di mettere le nostre mani al codice occorre tuttavia procurarsi le chiavi per entrambe le operazioni. Per ottenerle basta cliccare in alto a destra sul simbolo delle impostazioni (la rotellina dentata) vicino a *Quick Tests*.

A questo punto potete finalmente passare a creare la vostra applicazione iniziando dal codice per il Training. Per crearla aprite Visual Studio, *Nuovo Progetto*, *.Net Core*, *Console Application (.Net Core)*.

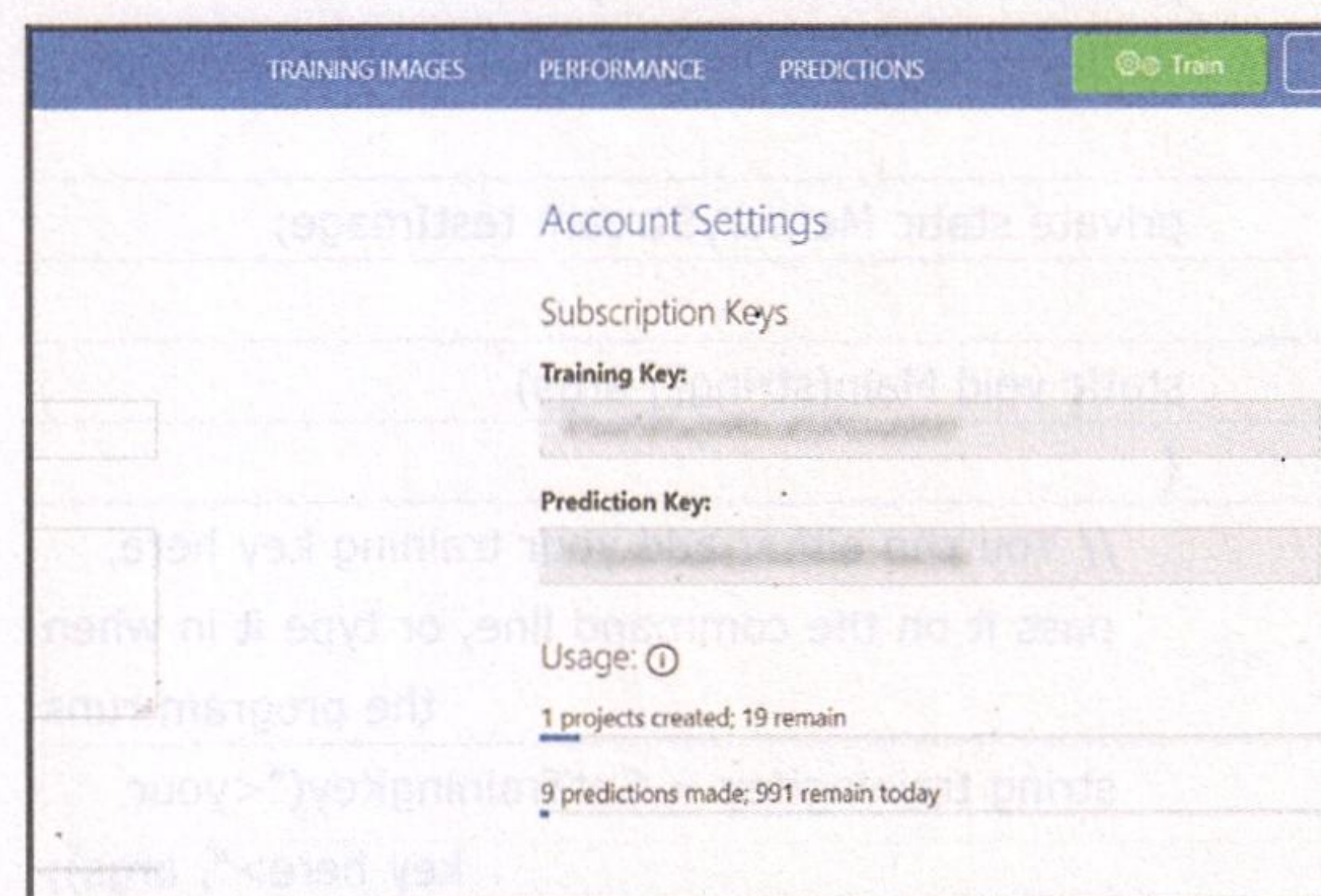
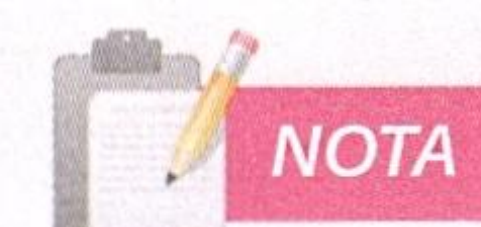



Fig. 17: Le chiavi necessarie ad usare il servizio

METTETEVI COMODI

Su channel9, il canale education di Microsoft, è disponibile, in lingua inglese, la presentazione delle Custom Vision API alla conferenza Build 2017: https://channel9.msdn.com/Events/Build/2017/T6022

Creato il progetto occorre aggiungete i riferimenti alla libreria che vi permetterà di utilizzare il servizio. Su Esplora risorse cliccate col tasto destro su *riferimenti*, quindi *Gestisci pacchetti Nuget…*

Nell'area di ricerca *Sfoglia* cercate *custom cognitive vision* quindi selezionate *Microsoft.Cognitive.CustomVision.Training*.

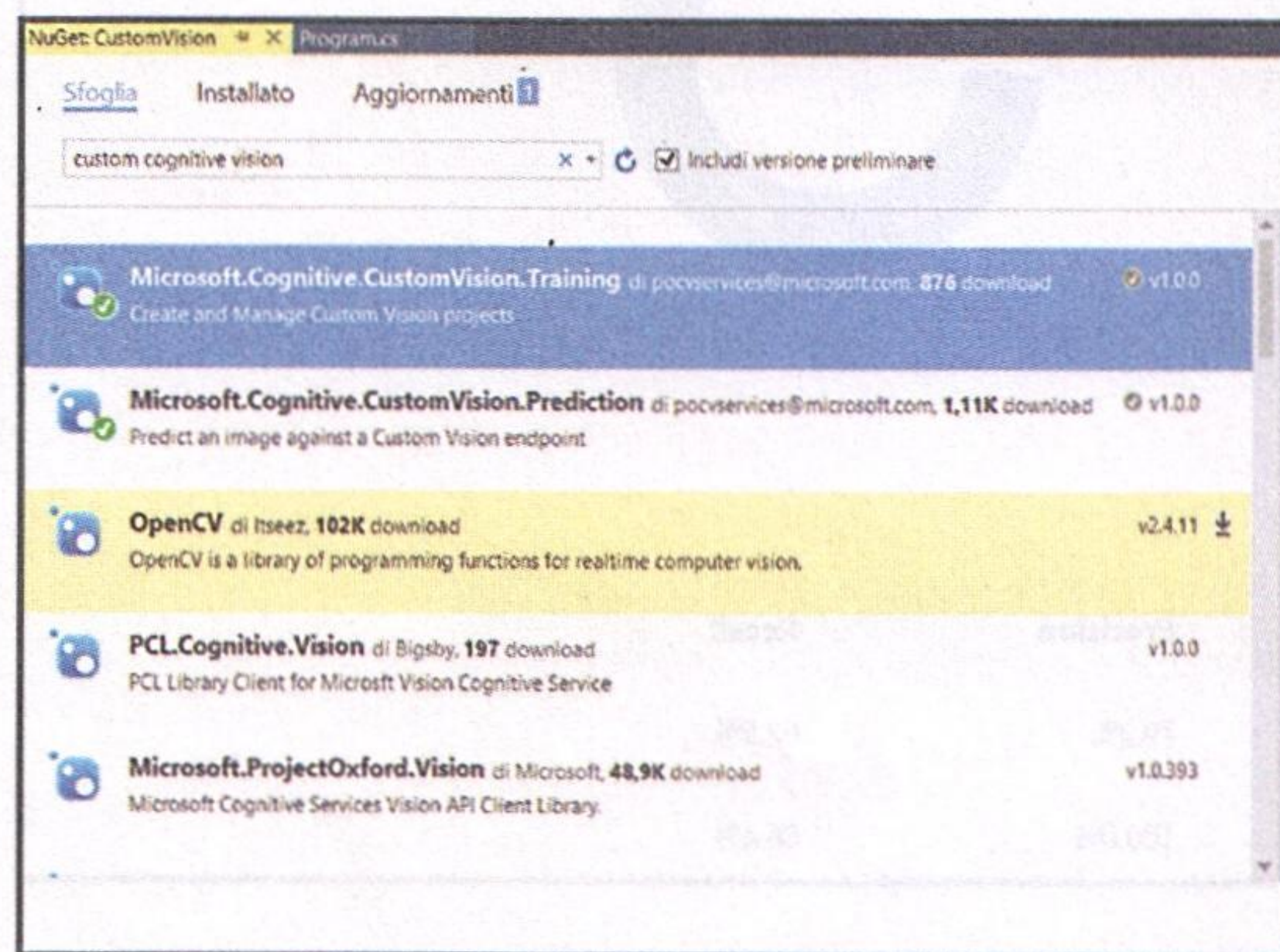



Fig. 18: La libreria di Microsoft che ci permette di utilizzare il servizio in addestramento

Il codice è il seguente:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading;
using Microsoft.Cognitive.CustomVision;

namespace DogTrainer
{
  class Program
  {
    private static List<MemoryStream>
                    barboncinoImages;
    private static List<MemoryStream>
                    labradorImages;
    private static List<MemoryStream>
                    pastore_tedescoImages;
    private static List<MemoryStream>
                    rottweilerImages;

    private static MemoryStream testImage;

    static void Main(string[] args)
    {
      // You can either add your training key here,
      pass it on the command line, or type it in when
                    the program runs
      string trainingKey = GetTrainingKey("<your
                    key here>", args);

      // Create the Api, passing in a credentials
                    object that contains the training key
      TrainingApiCredentials trainingCredentials =
                    new TrainingApiCredentials(trainingKey);
      TrainingApi trainingApi = new TrainingApi
                    (trainingCredentials);

      // Upload the images we need for training and
                    the test image
      Console.WriteLine("\tUploading images");
      LoadImagesFromDisk();

    // Create a new project
      Console.WriteLine("Creating new project:");
      var project = trainingApi.CreateProject("My
                    New Project");
    }

    private static string GetTrainingKey(string
                    trainingKey, string[] args)
    {
      if (string.IsNullOrWhiteSpace(trainingKey) ||
          trainingKey.Equals("<your key here>"))
      {
        if (args.Length >= 1)
        {
          trainingKey = args[0];
        }

        while (string.IsNullOrWhiteSpace
          (trainingKey) || trainingKey.Length != 32)
        {
          Console.Write("Enter your training key:
                    ");
          trainingKey = Console.ReadLine();
        }
        Console.WriteLine();
      }

      return trainingKey;
    }

    private static void LoadImagesFromDisk()
    {
      // this loads the images to be uploaded from
                    disk into memory
      barboncinoImages = Directory.
        GetFiles(@"..\..\..\..\..\Immagini_Addestramento\
        Barboncino").Select(f => new MemoryStream(File.
                    ReadAllBytes(f))).ToList();
      labradorImages = Directory.
        GetFiles(@"..\..\..\..\..\Immagini_Addestramento\
        Labrador").Select(f => new MemoryStream(File.
                    ReadAllBytes(f))).ToList();
      pastore_tedescoImages = Directory.
        GetFiles(@"..\..\..\..\..\Immagini_Addestramento\
        Pastore Tedesco").Select(f => new MemoryStream(File.
                    ReadAllBytes(f))).ToList();
      rottweilerImages = Directory.
        GetFiles(@"..\..\..\..\..\Immagini_Addestramento\ù
        Rottweiler").Select(f => new MemoryStream(File.
                    ReadAllBytes(f))).ToList();
```

```
            testImage = new MemoryStream(File.
    ReadAllBytes(@"..\..\..\..\..\Immagini_Addestramento\
                                   Test\test_image.jpg"));
        }
    }
}
```

Il flusso è abbastanza lineare, vengono create le liste di MemoryStream delle immagini da usare come addestramento:

- barboncinoImages
- labradorImages
- pastore_tedescoImages
- rottweilerImages

Inizializzate le chiavi vengono caricate massivamente le immagini con la funzione *LoadImagesFromDisk()*. In quest'ultima cambiate ovviamente i path di conseguenza. Infine viene creato un nuovo progetto con *trainingApi.CreateProject*.

Per creare nuovi tag, bastano le seguenti righe di codice:

```
var labradorTag = trainingApi.CreateTag(project.Id,
                                        "Labrador");
var pastore_tedescoTag = trainingApi.CreateTag(project.
                              Id, "Pastore Tedesco");
```

Per caricare le immagini sul server uno alla volta:

```
foreach (var image in hemlockImages)
{
    trainingApi.CreateImagesFromData(project.Id, image,
     new List<string>() { barboncinoTag.Id.ToString() });
}
```

Oppure in modalità bulk:

```
trainingApi.CreateImagesFromData(project.Id,
  barboncinoImages, new List<Guid>() { barboncinoTag.
                                              Id });
```

Per effettuare il training infine:

```
Console.WriteLine("\tTraining");
var iteration = trainingApi.TrainProject(project.Id);

while (iteration.Status == "Training")
{
    Thread.Sleep(1000);

    iteration = trainingApi.GetIteration(project.Id,
                                     iteration.Id);
}

iteration.IsDefault = true;
trainingApi.UpdateIteration(project.Id, iteration.Id,
                                        iteration);
Console.WriteLine("Done!\n");
```

Per effettuare la *Prediction*, ovvero utilizzare materialmente il classificatore occorre prima aggiungere il riferimento al pacchetto *Microsoft.Cognitive.CustomVision.Prediction*. Per utilizzare il servizio, finalmente:

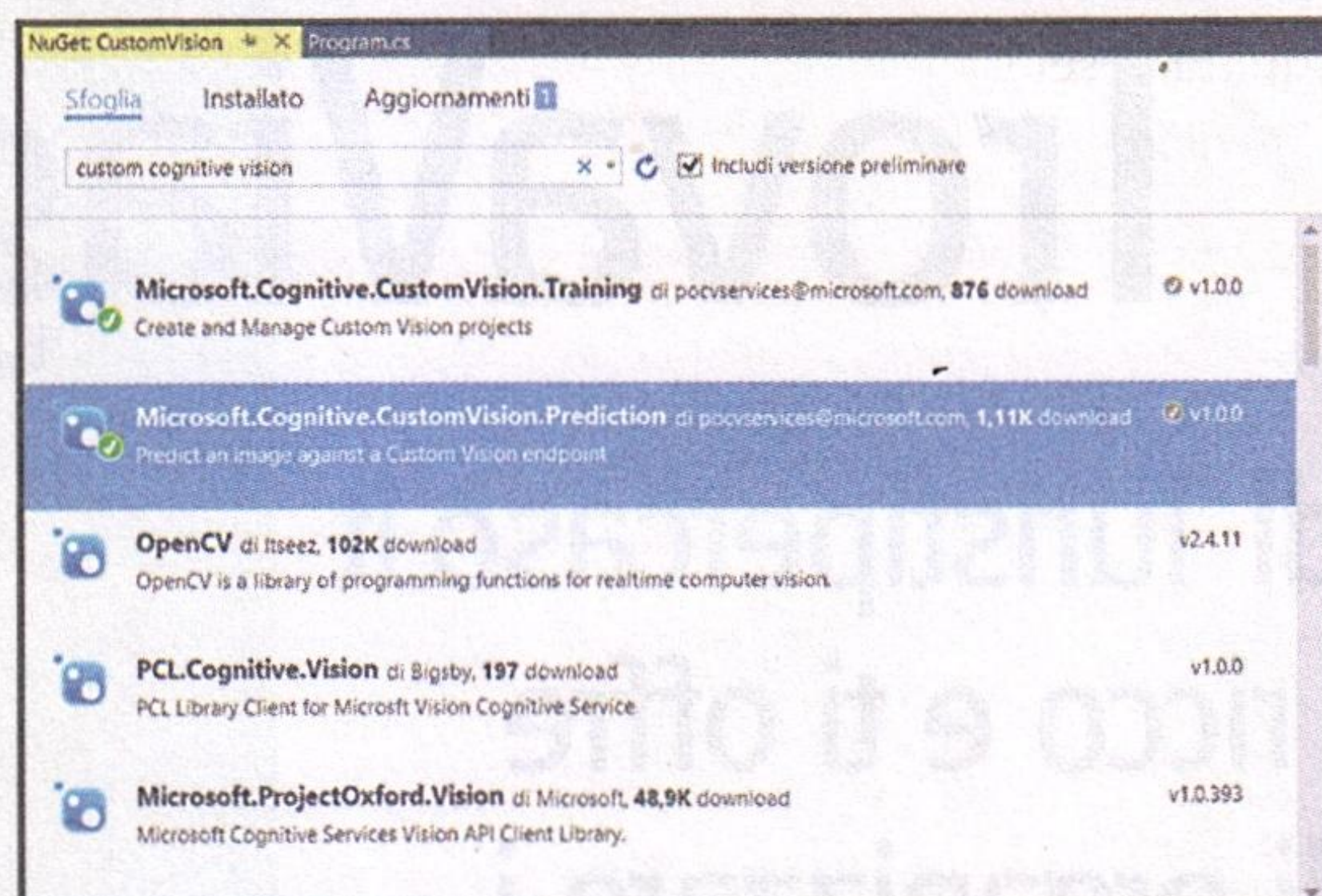



Fig. 19: La libreria di Microsoft che ci permette di utilizzare il servizio

```
var account = trainingApi.GetAccountInfo();
var predictionKey = account.Keys.PredictionKeys.
                                      PrimaryKey;

PredictionEndpointCredentials predictionEndpoint
      Credentials = new PredictionEndpointCredentials
                                   (predictionKey);
PredictionEndpoint endpoint = new PredictionEndpoint
                     (predictionEndpointCredentials);

Console.WriteLine("Making a prediction:");
var result = endpoint.PredictImage(project.Id,
                                       testImage);

foreach (var c in result.Predictions)
{
   Console.WriteLine($"\t{c.Tag}: {c.Probability:P1}");
}

Console.ReadKey();
```

CONCLUSIONI

Non è un mistero che Microsoft sta spingendo moltissimo sul tema intelligenza artificiale. I passi avanti sono notevoli rispetto a solo un anno fa ed in alcuni campi ha addirittura segnato record storici. Ma ispirandosi ad una nota pubblicità di qualche anno fa: "La potenza è nulla senza controllo". Le Custom Vision API ne incarnano la filosofia. Di soluzioni estremamente performanti ma difficili da utilizzare ne è pieno il panorama attuale, rendere questo potenziale a portata di tutti, in termini di facilità di utilizzo, è da pochi. Non ci sono quindi scuse: come abbiamo visto con poche righe di codice si ha a disposizione tutta la potenza delle reti neurali di ultima generazione. Non vi resta quindi che mettervi al lavoro e sprigionare tutto il potenziale a disposizione!

Sebastiano Galazzo

L'AUTORE

Sebastiano Galazzo (@galazzoseba su Twitter!) Microsoft MVP per la categoria Artificial Intelligence. Appassionato di Intelligenza Artificiale, sviluppa da anni algoritmi legati a motori di ricerca semantici, ed elaborazione delle immagini. Il know how su questi temi gli ha permesso vincere diversi premi nazionali ed internazionali. Può essere contattato all'indirizzo sebastiano.galazzo@gmail.com.

IL BOT PARLA COME TE!

PER MIGLIORARE LE RELAZIONI TRA GLI UTENTI E I BOT CHE RISPONDONO AUTOMATICAMENTE È NECESSARIO CHE I BOT PARLINO CON UN LINGUAGGIO SIMILE A QUELLO UMANO. LE LIBRERIE CHATTERBOT E TELEPOT CI PERMETTONO DI COSTRUIRE UN BOT "UMANOIDE" PER TELEGRAM

La strada verso la domotica, il nuovo mercato dell'informatica che nel giro dei prossimi 10 anni si svilupperà in modo esponenziale, è costruita da tante tecnologie che oggi esistono già, ma che devono essere migliorate e unite ad altre. Ovviamente le reti neurali sono un componente fondamentale: mettere un computer a controllare casa propria ha senso se questo computer può essere intelligente. Il vantaggio dell'intelligenza artificiale e del meccanismo di machine learning (cioè della capacità dei computer di imparare da esempi) è la possibilità di rendere i computer molto più "umani" nel modo di rapportarsi. Non è un caso che il sogno di un qualsiasi appassionato di informatica sia avere una sorta di Jarvis di Iron Man in casa propria, un computer che non soltanto gestisce la casa, ma che parla e ascolta, rispondendo come farebbe un essere umano. Oggi noi abbiamo già delle tecnologie che permettono di ottenere dei buoni risultati, solo che non le utilizziamo ancora per la domotica. Anche perché spesso hanno un sorgente chiuso e alla gente non piace avere microfoni in casa che ascoltano qualsiasi cosa si dica senza sapere che succede ai propri dati. Per impieghi semplici, comunque, la tecnologia è già sul piatto e possiamo provare a migliorarla. In particolare, è possibile creare un programma capace di rispondere all'utente come farebbe una persona. È qualcosa che in parte siamo già abituati a vedere con gli smartphone, perché esistono degli assistenti vocali che rispondono all'utente utilizzando un linguaggio umano e non una rigida sintassi da linguaggio di programmazione. In questo articolo vogliamo proprio presentare la costruzione di un bot per Telegram che sia in grado di dialogare con l'utente. Cercheremo di mantenere il progetto quanto più generico possibile, per rendere facile l'implementazione in diversi contesti (un semplice bot "amico" con cui chiacchierare, un robot "maggiordomo", una centralina domotica per la casa, eccetera…) ma ciò che presentiamo è un progetto immediatamente funzionante.

LA CLASSE PRIMA DI TUTTO

La parte di dialogo verrà gestita tramite una libreria apposita chiamata Chatterbot: è una libreria Python completamente open source che permette di realizzare molto facilmente dei bot in grado di rispondere a frasi in qualsiasi lingua. La libreria utilizza un set di

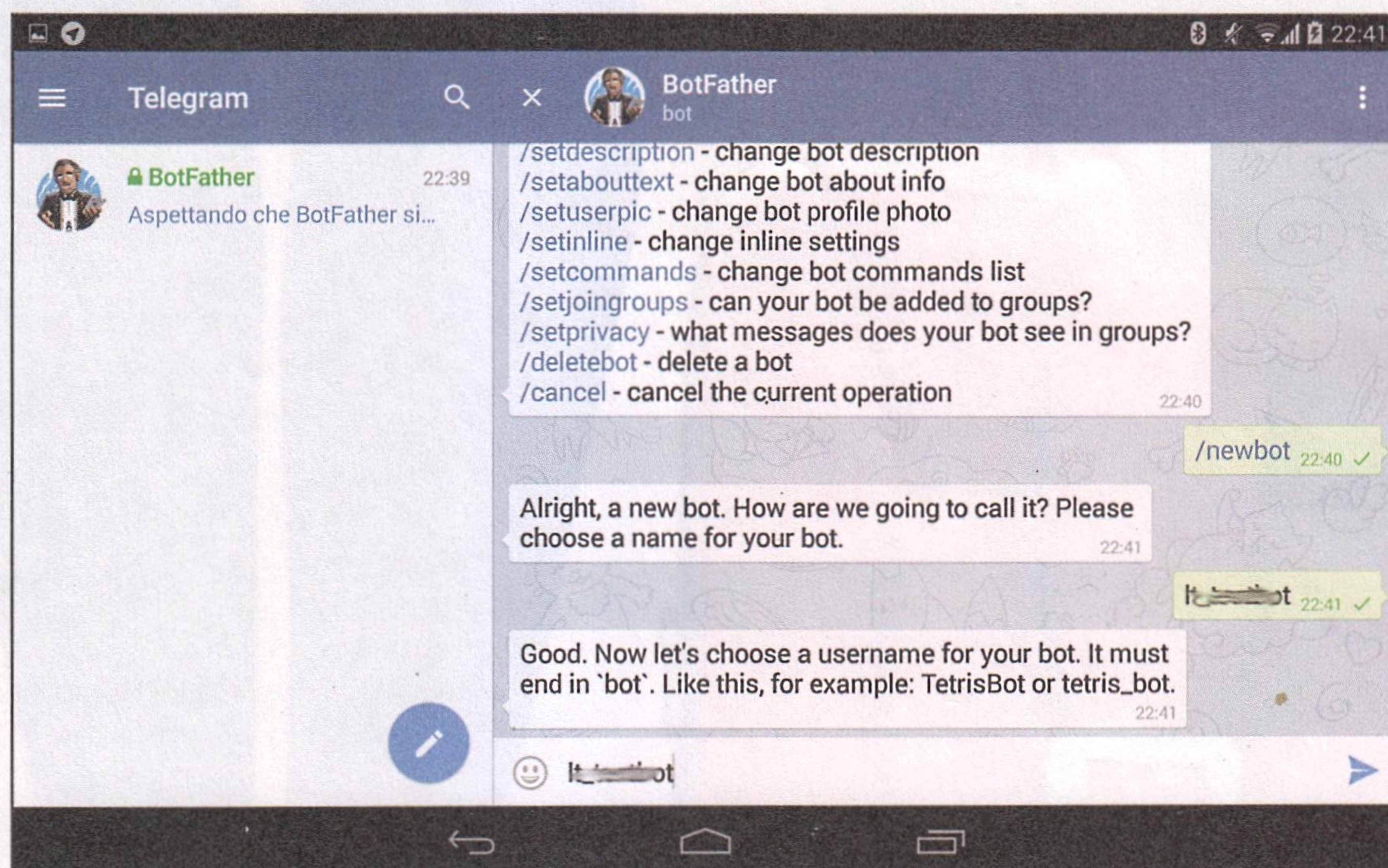



Fig. 1: Per creare un nuovo bot Telegram basta dare il comando /newbot al bot BotFather

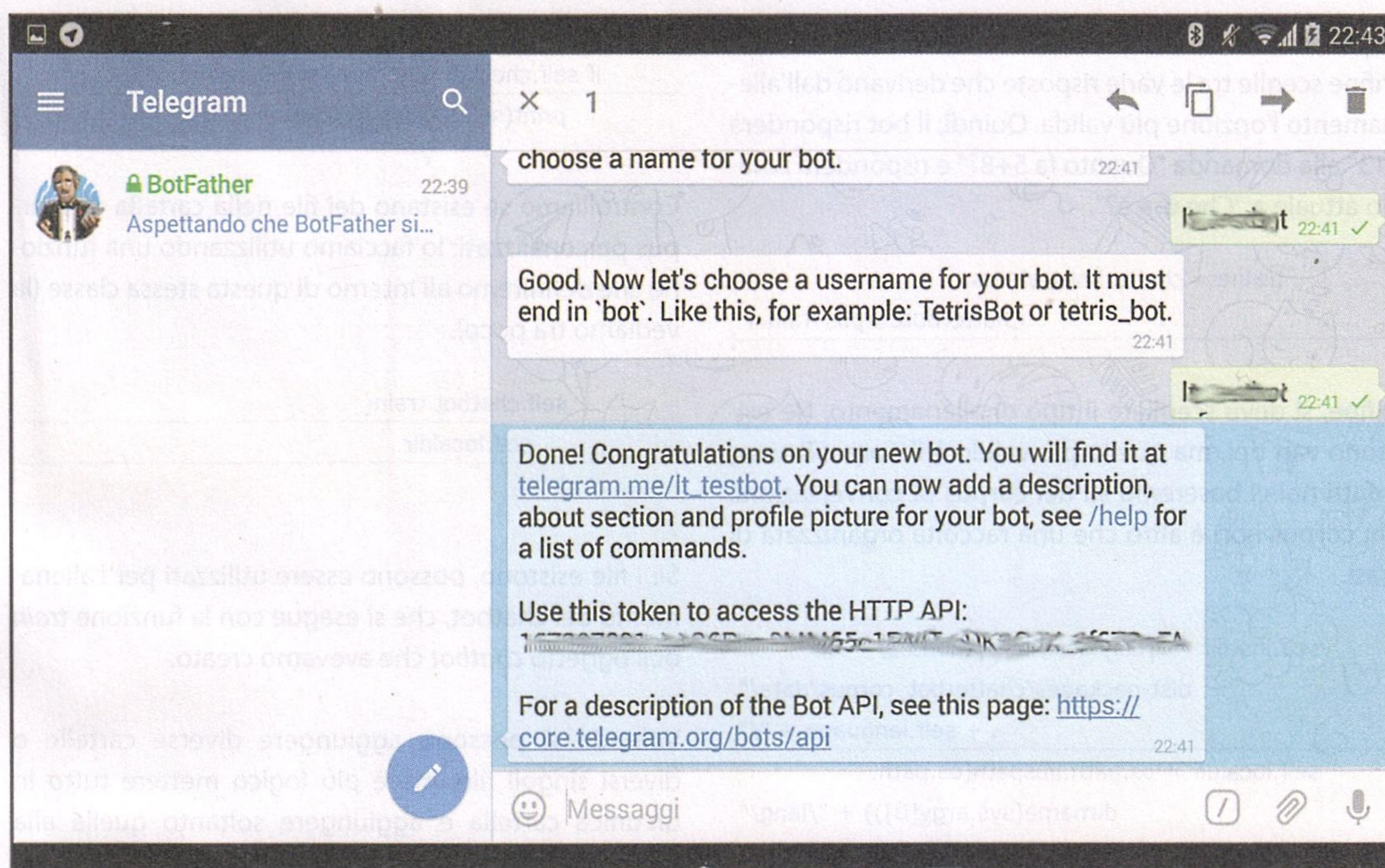



Fig. 2: BotFather fornisce il codice token del proprio bot appena creato

conversazioni di esempio dalle quali imparare: non dobbiamo fare altro che trascrivere delle conversazioni e il bot le utilizzerà per capire con quali parole rispondere nel momento in cui riceve un certo messaggio.
La libreria è abbastanza facile da utilizzare, ma richiede una certa configurazione: per assicurarci di avere le stesse impostazioni in tutti i casi (potremmo voler utilizzare chatterbot anche in altre situazioni) costruiremo una apposita classe, che potremo riutilizzare in qualsiasi progetto futuro senza necessità di modifiche.
Cominciamo quindi a scrivere il file *zorbachatter.py*:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
```

Il file sarà basato su Python3, quindi lo definiamo con la classica riga *shebang*.

```
import os
import os.path
import sys
from chatterbot import ChatBot
```

Le librerie da importare sono poche, soltanto quelle relative al sistema operativo per controllare i file. Si deve poi aggiungere la libreria *ChatterBot*, in particolare il suo elemento *ChatBot*.

```
class ZorbaChatter(object):
```

La definizione della classe è molto semplice, basta indicare il suo nome (che abbiamo deciso essere *ZorbaChatter*, ma potete cambiarlo).

```
    def __init__(self, lang = "english"):
```

Il costruttore della classe è la funzione init, che permette di seguire una serie di operazioni nel momento in cui si comincia ad utilizzare la classe stessa in un altro programma. Come argomenti richiediamo soltanto la lingua in cui il bot dovrà parlare, che comunque impostiamo a *english* nel caso tale informazione non venga specificata manualmente.

```
        self.language = lang
```

Impostiamo il linguaggio in una variabile: affinché tale variabile sia accessibile anche dalle altre funzioni della classe dobbiamo assegnarla all'oggetto *self*, che indica proprio la classe attuale.

```
        self.chatbot = ChatBot(
        'Zorba',
```

Cominciamo ora a definire il chatbot che vogliamo realizzare. Quando si dichiara un oggetto di tipo *ChatBot* si devono indicare una serie di impostazioni. La prima è ovviamente il nome, che può essere qualsiasi cosa si desideri.

```
            logic_adapters=[
            "chatterbot.logic.MathematicalEvaluation",
            "chatterbot.logic.TimeLogicAdapter",
            "chatterbot.logic.BestMatch"
            ],
```

Sono poi necessari degli adattatori logici, cioè delle modalità di manipolazione dei messaggi in base a semplici regole. L'ordine in cui questi vengono inseriti è importante: secondo quello che abbiamo scritto, il bot prima di tutto cercherà di capire se nel messaggio vi siano delle espressioni matematiche, poi cerca di

CREARE UN BOT SU BOTFATHER

Per realizzare un nuovo bot, è necessario ottenere l'autorizzazione da BotFather. Serve prima di tutto un accesso a Telegram (se non si vuole utilizzare uno smartphone si può accedere da PC all'indirizzo https://web.telegram.org/). Dobbiamo aprire Telegram e cercare l'utente BotFather: quando lo troviamo, iniziamo una nuova conversazione con tale utente. BotFather, ci viene spiegato nel suo primo messaggio, è a sua volta un bot che risponde a particolari comandi. Il primo comando che dobbiamo dare è **/newbot**, ed indica che vogliamo costruire un nuovo bot.

capire se ci siano delle richieste relative all'orario, e infine sceglie tra le varie risposte che derivano dall'allenamento l'opzione più valida. Quindi, il bot risponderà "13" alla domanda "Quanto fa 5+8?" e risponderà l'orario attuale a "Che ora è?".

```
        trainer='chatterbot.trainers.
                        ChatterBotCorpusTrainer'
    )
```

Infine, si deve scegliere il tipo di allenamento. Ne esistono vari tipi, ma quello più valido è il *CorpusTrainer*, infatti noi ci baseremo su dei corpus di conversazioni. Un corpus non è altro che una raccolta organizzata di frasi.

```
    self.instdir = "/usr/local/lib/python3.5/
                dist-packages/chatterbot_corpus/data/"
                            + self.language + "/"
    self.localdir = os.path.abspath(os.path.
                    dirname(sys.argv[0])) + "/lang/"
                + self.language + "/chatbotcorpus/"
```

Il punto, ora, è trovare i file dei corpus da caricare. Definiamo quindi almeno due posizioni: una rappresenta la cartella di installazione della libreria *chatterbot* (su un sistema Raspbian è quella che abbiamo indicato, su Windows è ovviamente diversa). L'altra rappresenta la cartella nella quale vogliamo inserire i nostri corpus personalizzati (che comincia dalla posizione in cui si trova l'attuale file, e sarà qualcosa del tipo **/lang/italiano/chatbotcorpus/**).

COMINCIARE L'ALLENAMENTO

Cominciamo ora a scrivere la funzione che si occuperà dell'allenamento del bot.

```
def train(self):
    if self.checkdirnotempty(self.localdir):
        print(self.localdir)
```

Controlliamo se esistano dei file nella cartella dei corpus personalizzati: lo facciamo utilizzando una funzione che definiremo all'interno di questa stessa classe (la vediamo tra poco).

```
        self.chatbot.train(
            self.localdir
        )
```

Se i file esistono, possono essere utilizzati per l'allenamento del chatbot, che si esegue con la funzione *train* dell'oggetto *chatbot* che avevamo creato.

Volendo si possono aggiungere diverse cartelle o diversi singoli file, ma è più logico mettere tutto in un'unica cartella e aggiungere soltanto quella alla funzione train.

```
    elif self.checkdirnotempty(self.instdir):
        print(self.instdir)
        self.chatbot.train(
            self.instdir
        )
```

Ovviamente, è possibile che non abbiamo sviluppato nessun corpus personalizzato per il programma che scriviamo al momento: in tal caso si ripiega sulla cartella installata dallo stesso chatterbot.

```
    else:
        print("Using standard english corpus")
        self.chatbot.train("chatterbot.corpus.
                        english.greetings")
```

NOTA

PROVARE IL PROGETTO

BotFather ci chiederà di specificare il nome del nostro bot: possiamo chiamarlo come vogliamo, l'importante è che il nome sia composto da una parola sola (si possono unire più parole utilizzando i simboli – oppure _ ma non lo spazio). Adesso è necessario indicare un nome per l'utente che rappresenta il bot: la regola è che questo nome deve terminare con le lettere "bot". Per esempio, provabot, oppure prova_bot. Per il resto, possiamo utilizzare qualsiasi carattere, esclusi gli spazi: anche il nome dell'utente deve essere un unica parola. Alla fine, BotFather ci fornisce il token del nostro bot.

Fig. 3: È necessario modificare i file telegramtoken.txt e botusers.csv inserendo il token del proprio bot e il proprio ID utente di Telegram

È possibile che non esista nemmeno la cartella dei corpus installati da chatterbot, almeno per la lingua che è stata scelta. In tal caso chiamiamo la funzione *train* di chatbot utilizzando un corpus standard per l'inglese, che è certamente installato.

```
def reply(self, phrase = ""):
        response = self.chatbot.get_response(phrase)
        return response
```

Ovviamente si deve anche permettere l'effettiva funzionalità del bot, cioè a frase deve rispondere con un'altra frase. Lo facciamo implementando la funzione *reply*, che si occupa semplicemente di sfruttare il metodo *get_response* per ottenere dal chatbot una risposta alla frase dell'utente.

```
def checkdirnotempty(self, folder = ""):
        check = False
        if os.path.isdir(folder):
            entities = os.listdir(folder)
            for entity in entities:
                if os.path.isfile(folder + entity):
                    check = True
                    break
        return check
```

La funzione, per il controllo dell'esistenza dei file del corpus di allenamento è abbastanza semplice: fornisce un valore *False* se la cartella specificata non contiene alcun file, e un valore *True* se invece vi sono dei file. Prima di tutto verifica che la cartella esista e poi cerca un elenco dei file per verificare che ne esista almeno uno. La classe è terminata, ora si può procedere a lavorare direttamente sul file del bot Telegram.

IL BOT

Per il bot vero e proprio creiamo un altro file, cominciando sempre con l'importazione di alcune librerie:

```
import time
import datetime
import telepot
import os
import sys
from zorbachatter import ZorbaChatter
```

Tra le librerie standard è ovviamente fondamentale Telepot, quella che permette la costruzione di bot in Python per Telegram. Poi si deve aggiungere la classe che abbiamo appena creato. Il file *zorbachatter.py* deve trovarsi nella stessa cartella del programma attuale, e deve contenere la classe *ZorbaChatter*. Se avete usato nomi diversi, li dovete sostituire.

```
telegramtoken = ''
checkuserid = 1
usersfile = 'botusers.csv'
attemptsfile = '/tmp/attempts.log'
active = 1
```

Definiamo alcune variabili fondamentali: il token di telegram, che conviene lasciare per il momento vuoto, una variabile per stabilire se si debbano controllare gli utenti che accedono al bot e un paio di file per memorizzare sia gli utenti che i tentativi di accesso non autorizzati. La variabile *active* ci permette di attivare o disattivare il bot anche a runtime.

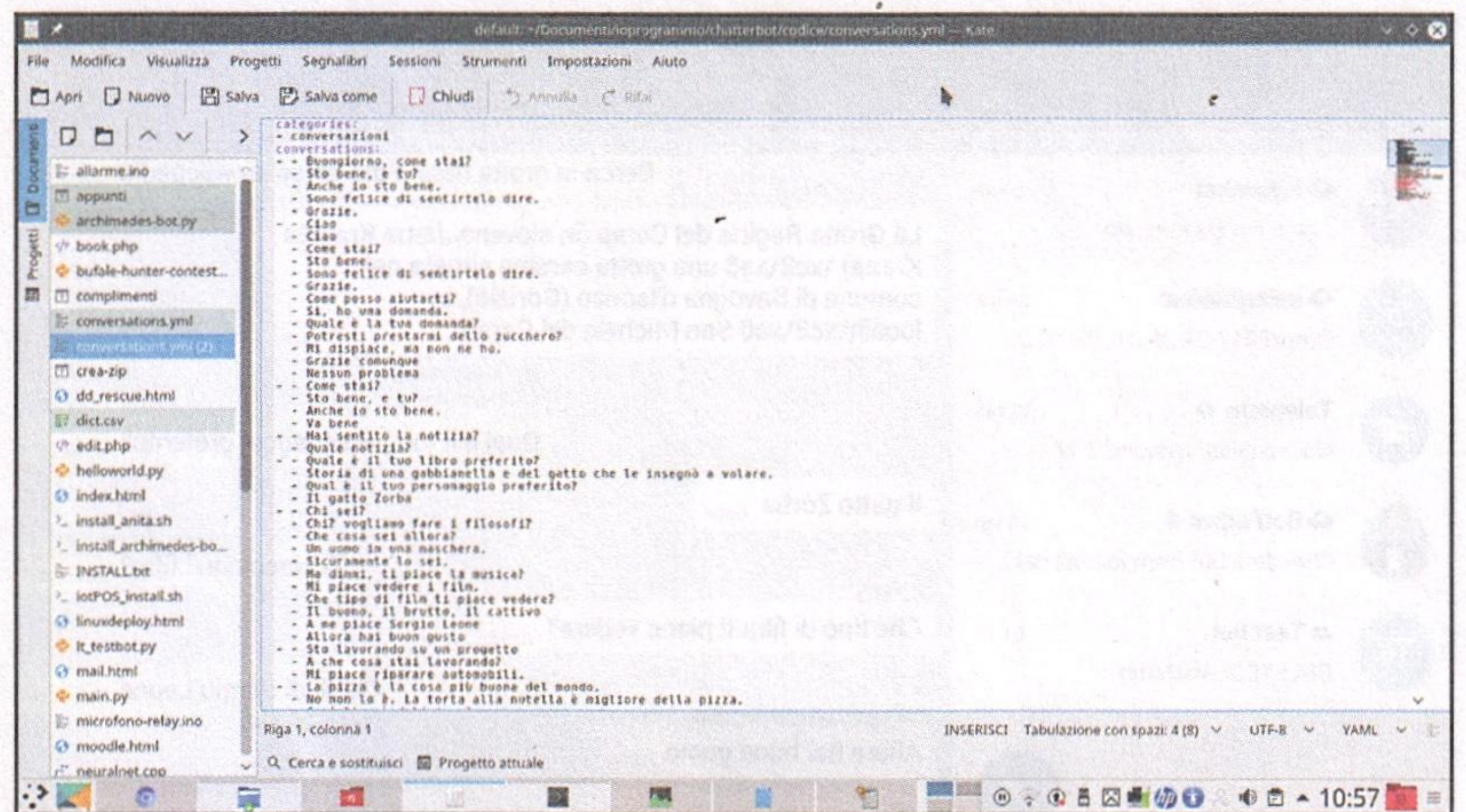



Fig.4: Il file conversations.yml contiene gli esempi delle conversazioni da cui il bot può imparare

```
language = "it-IT"
```

Ora definiamo la lingua con cui vogliamo lavorare: questo sarà il nome della cartella in cui abbiamo memorizzato il corpus da caricare nel chatterbot (nell'esempio è *it-IT* ma potrebbe essere *italian* o *italiano*, a seconda di quello che avete scelto).

```
chatter = ZorbaChatter(language)
chatter.train()
```

Ora, infatti, creiamo l'oggetto *chatter* utilizzando la classe *ZorbaChatter* creata poco fa.

Creiamo l'oggetto adesso e iniziamo l'allenamento con la funzione *train()* perché così questo allenamento (che è molto lento) avviene soltanto adesso e non ogni volta che il bot deve rispondere ad un messaggio.

```
if telegramtoken == '' and os.path.isfile
                                ("telegramtoken.txt"):
    text_file = open("telegramtoken.txt", "r")
    telegramtoken = text_file.read().replace("\n", "")
    text_file.close()
```

Nel caso in cui il token del bot di Telegram non sia ancora stato specificato, lo aggiungiamo andandolo a leggere da un file: questa è l'opzione migliore, perché

PROVARE IL PROGETTO

Per provare il programma che presentiamo basta ottenere il token del proprio bot tramite BotFather e memorizzare tale testo nel file *telegramtoken.txt*, nella stessa cartella del programma. Poi si devono installare i pacchetti, cosa che si può fare con PIP usando questi comandi:
pip3 install telepot
pip3 install chatterbot
Infine, è necessario creare, sempre nella stessa cartella del programma, il file *botusers.csv* inserendo al suo interno il proprio ID utente di Telegram. Se non conoscete il vostro ID utente ve lo dirà il bot stesso appena provate a contattarlo (prima di avere creato il file *botusers.csv*).

così se si vuole distribuire il proprio codice sorgente non si rischia di mostrare a tutti anche il proprio token (basta non pubblicare il file telegramtoken).

```
print("Connecting to Telegram...")
```

Fig.5: Il nostro bot risponde a domande di carattere generale usando un linguaggio umano

```
bot = telepot.Bot(telegramtoken)
print(bot.getMe())
```

Ora si può creare il bot utilizzando la libreria *telepot*. Non bisogna fare altro che specificare il token di Telegram.

LEGGERE I MESSAGGI

La funzione che si occuperà di gestire i messaggi ricevuti dal bot è chiamata *handle* e prende in oggetto il messaggio ricevuto.

```
def handle(msg):
    global bot
    global chatter
    global language
```

Ci servono alcuni elementi dalla routine principale del programma: *bot, chatter, language*.

```
    chat_id = msg['chat']['id']
    sender = msg['from']['id']
```

Preleviamo dal messaggio due informazioni fondamentali: l'ID della chat in corso e quello del mittente del messaggio.

```
    users = listusers()
```

Otteniamo la lista degli utenti autorizzati a parlare con il bot dalla funzione *listusers*. È una funzione molto semplice che non descriviamo qui per motivi di spazio ma che trovate nel codice sorgente.

```
    if checkuserid == 1:
        verified = 0
        if users != "":
            for usr in users:
                if str(sender) == usr:
                    verified = 1
```

Se il bot deve verificare l'identità degli utenti, si utilizza un semplice ciclo for per scorrere la lista di utenti autorizzati e verificare se l'attuale utente sia effettivamente inserito nell'elenco.

```
        if verified == 0:
            bot.sendMessage(chat_id, "I don't talk with
                            strangers, dear "+str(sender))
            return
```

Se l'utente non è stato verificato, gli si invia un messaggio che specifica che non è autorizzato a parlare con il bot, e la funzione viene interrotta dal comando *return*.

```
    command = ''
    try:
        if msg['text'] != '':
            command = msg['text']
            print('Got command: ' + command)
    except:
        print("No text in this message")
```

Supponendo quindi che l'utente sia stato verificato, possiamo procedere ad analizzare il suo messaggio. L'eventuale comando da eseguire, o il testo a cui rispondere, sarà contenuto proprio nell'oggetto *msg* che ci viene fornito da Telegram, alla voce *text*. Quindi se esiste assegniamo il testo del messaggio alla variabile *command*. L'operazione è inserita in un blocco *try-except* perché è possibile che il messaggio inviato non contenga alcun testo (potrebbe essere una immagine o un audio).

```
    if command == '/time':
        bot.sendMessage(chat_id, str(datetime.
                        datetime.now()))
```

Se il messaggio conteneva del testo, possiamo cominciare ad analizzarlo. Per esempio, è possibile controllare se specifiche parole siano presenti all'interno del messaggio per rispondere in un certo modo. Se il messaggio contiene la parola */time*, rispondiamo all'utente indicando la data ed ora attuali. Utilizziamo lo slash iniziale per evitare che il programma possa confondersi con la parola time utilizzata in qualche frase.

```
    elif '/adduser' in command:
        if len(command.split(' ')) > 1:
```

NOTA

DOVE PROCURARSI UN RASPBERRY

Ormai, molti negozi di elettronica ed addirittura di modellismo vendono sia Arduino che RaspberryPi. I rivenditori ufficiali, comunque, sono Farnell ed RSComponent, ma anche Amazon Italia. È poi necessario avere un alimentatore microUSBB da 2 A (10 Watt). Infatti, gli alimentatori da 1 A non forniscono abbastanza corrente per supportare le varie porte USB del Raspberry. Attualmente sono disponibili tre diversi modelli: Raspberry2, Raspberry3, e Raspberry Zero. Tuttavia, il Raspberry Zero è progettato per l'uso come dispositivo indossabile (wearable), ha poca potenza come server. Il Raspberry 2 ed il 3 sono molto simili, dal puntò di vista server: il 3 ha dei vantaggi, come il WiFi integrato, ma produce più calore ed è necessario usare delle piastrine di dissipazione per il processore.

```
            usrname = command.split(' ')[1]
            adduser(usrname)
            bot.sendMessage(chat_id, "User "+
                                        usrname+" added")
    elif '/deluser' in command:
        if len(command.split(' ')) > 1:
            usrname = command.split(' ')[1]
            deluser(usrname)
            bot.sendMessage(chat_id, "User "+usrname+"
                                        deleted")
```

In modo analogo, possiamo implementare altri comandi. Per esempio, uno per aggiungere ed uno per rimuovere utenti dalla lista di quelli autorizzati. Anche in questo caso trovate le relative funzioni nel codice sorgente completo, si vede però che è possibile manipolare il messaggio dell'utente come una qualsiasi altra stringa di testo, in questo caso estraendo il testo dopo il primo spazio, testo che dovrebbe rappresentare l'id dell'utente da abilitare. La risposta all'utente avviene tramite la funzione *sendMessage* di telepot, che richiede in argomento l'id della chat e il messaggio da fornire.

```
    elif command == '/exit':
        global active
        active = False
        bot.sendMessage(chat_id, "The bot will shutdown
                                        in 10 seconds")
```

Un altro comando utile è */exit*, per impostare la variabile *active* al valore *False*, così il programma si interromperà. Questo ci permette di disattivare il bot inviandogli un messaggio: è una funzione che può essere utile se dovessero esserci problemi con il programma.

```
    elif command != '':
        answer = chatter.reply(command)
        bot.sendMessage(chat_id, str(answer))
```

Se nessuna di queste opzioni si è verificata, significa che il messaggio dell'utente non è affatto un comando, ma una semplice frase che richiede una risposta. Possiamo quindi procurarci una risposta invocando la funzione *reply* dell'oggetto *chatter*, che come abbiamo visto è la nostra classe che si interfaccia con ChatterBot. Possiamo poi fornire questa risposta all'utente con la solita funzione *sendMessage*, ma assicurandoci che la risposta *answer* venga gestita come una stringa (chatterbot, infatti, restituisce uno Statement, che non è un tipo di stringa riconosciuto da Telepot). Per fortuna basta il metodo *str()* per convertire lo Statement in una stringa.

```
bot.message_loop(handle)
print('I am listening ...')
```

Terminata la funzione, ricomincia la routine principale. Finora il bot non è stato in ascolto dei messaggi: per farlo ascoltare, si deve assegnare il nome dell'apposita funzione (che abbiamo chiamato *handle*) al message_loop del bot costruito con telepot.

```
while active:
    time.sleep(10)
print("Exiting")
sys.exit()
```

Naturalmente, se terminassimo qui la stesura del codice, il programma si interromperebbe. Per evitarlo utilizziamo un ciclo *while* controllato dalla variabile *active*, che quindi girerà in eterno finché tale variabile non verrà impostata a *False* con il comando */exit* che avevamo già definito. Il ciclo non fa altro che aspettare 10 secondi prima di ricominciare da capo. Mentre il ciclo è attivo il programma sarà in esecuzione, e mentre il programma è in esecuzione il *message_loop* di telepot continuerà ad intercettare i messaggi inviati al bot e passarli alla funzione *handle()*.

IL CORPUS DI ALLENAMENTO

La parte logica del nostro bot è stata costruita: il programma è pronto. Trattandosi di un sistema di machine learning, però, la qualità dei risultati dipenderà dall'allenamento che forniamo a ChatterBot. Quindi, dobbiamo costruire i file necessari per insegnare al bot come rispondere alle domande: in altre parole, i corpus di allenamento.

Il primo file da scrivere è *math_words.json*, che contiene la traduzione in formato numero dei numeri scritti a parole:

```
{
  "numbers" : {
    "uno" : 1,
    "due" : 2,
    "tre" : 3,
    "quattro" : 4,
    "cinque" : 5,
    "sei" : 6,
    "sette" : 7,
    "otto" : 8,
    "nove" : 9,
    "dieci" : 10
  },
```

Lo stesso file deve contenere anche la "traduzione" delle parole relative a operazioni e grandezze:

```
  "words" : {
    "più" : "+",
    "diviso" : "/",
    "meno" : "-",
    "per" : "*",
```

NOTA

DOCUMENTAZIONE UFFICIALE DI CHATTERBOT

Abbiamo descritto il funzionamento principale della libreria ChatterBot. Se tuttavia volete capire come funzioni più nel dettaglio, per personalizzare meglio il vostro bot, potete consultare la documentazione ufficiale online. Sono anche presenti degli esempi pratici che aiutato a comprendere l'utilizzo delle varie funzioni. Ad ogni modo, come in ogni progetto di machine learning, ricordiamo che i risultati dipendono soprattutto dai corpus per l'allenamento che si costruiscono, quindi è su quelli che è importante lavorare.
http://chatterbot.readthedocs.io/en/stable/

```
    "al quadrato" : "** 2",
    "alla potenza di" : "**"
  },
  "scales" : {
    "centinaia" : "* 100",
    "migliaia" : "* 1000",
    "milioni" : "* 1000000",
    "miliardi" : "* 1000000000"
  }
}
```

Un altro file molto importante, che non riportiamo per ovvii motivi, è *swear_words.csv*: contiene insulti e parole poco appropriate. Le parole presenti in questo file non verranno mai usate dal bot per comporre le proprie risposte, è un modo per evitare che l'apprendimento del bot possa degenerare nelle oscenità verbali. Poi si possono costruire tante conversazioni quante si vogliono, inserendole in diversi file per facilitare la lettura. Per esempio, si può creare il file *greetings.yml* per le tipiche battute di saluto:

```
categories:
- saluti
conversations:
- - Ciao
  - Ciao
- - Auguri!
  - Ciao
- - Ciao
  - Auguri!
- - Ciao, come va?
  - Bene
- - Ciao, come va?
- Va bene.
- - Sono felice di conoscerti.
  - Anche io ne sono felice
```

Come si può notare, ogni conversazione in questo file è rappresentata da due sole righe, la prima (quella che inizia la conversazione) è marcata dal simbolo - -, mentre la seconda (cioè la risposta che il bot dovrebbe dare alla prima) soltanto dal simbolo -.

CONVERSAZIONI PIÙ LUNGHE

Possiamo anche creare conversazioni più lunghe, come quelle nel file *conversations.yml*:

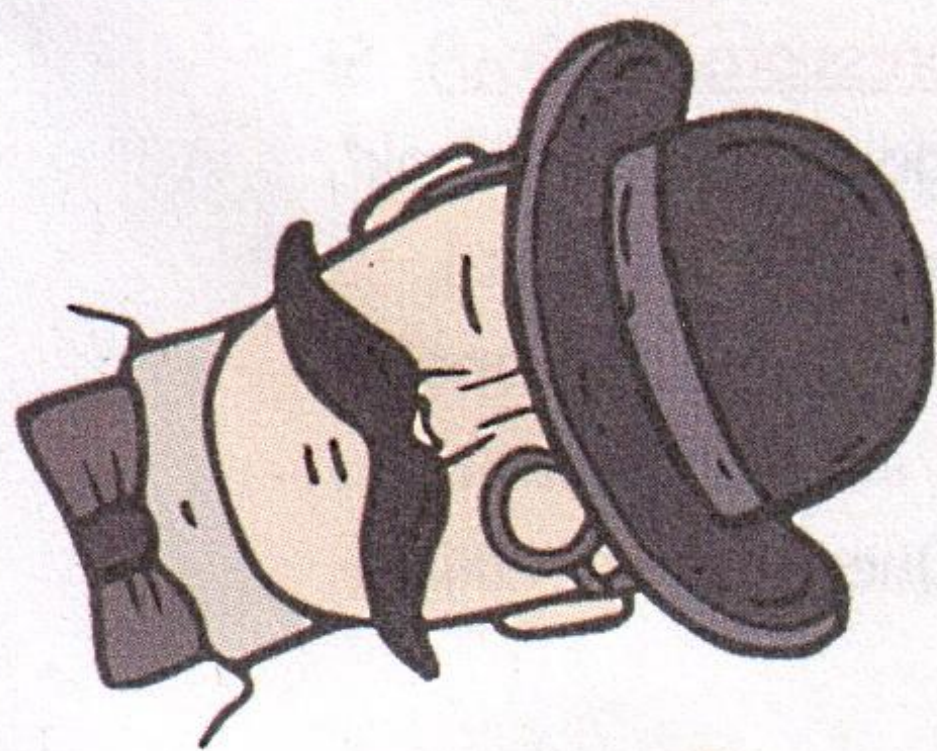
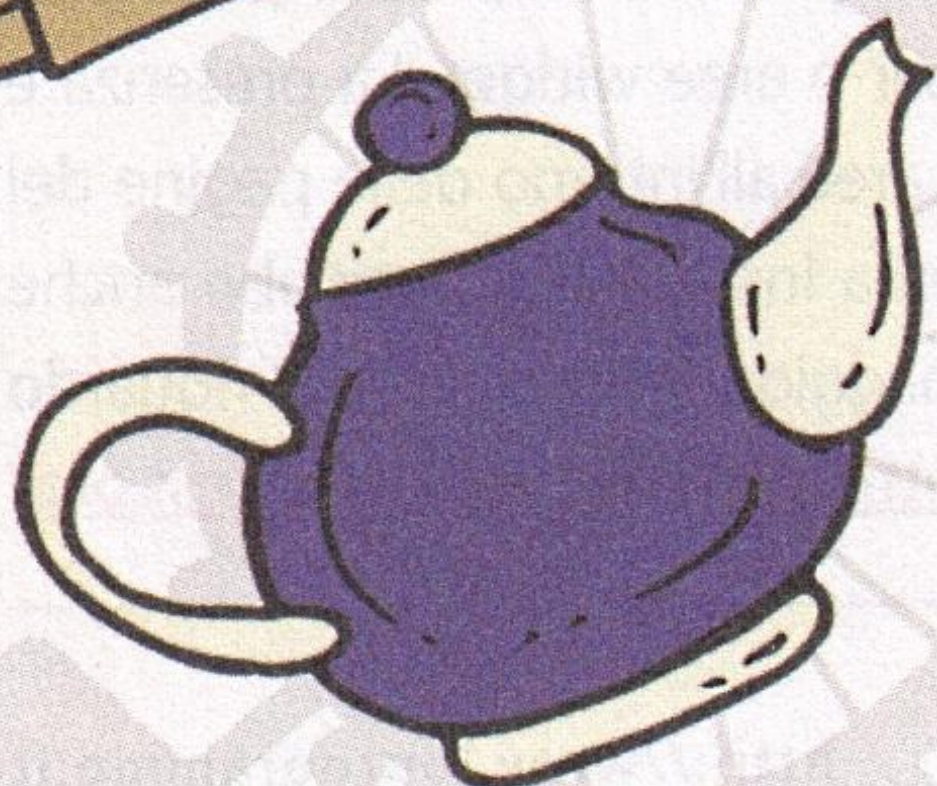
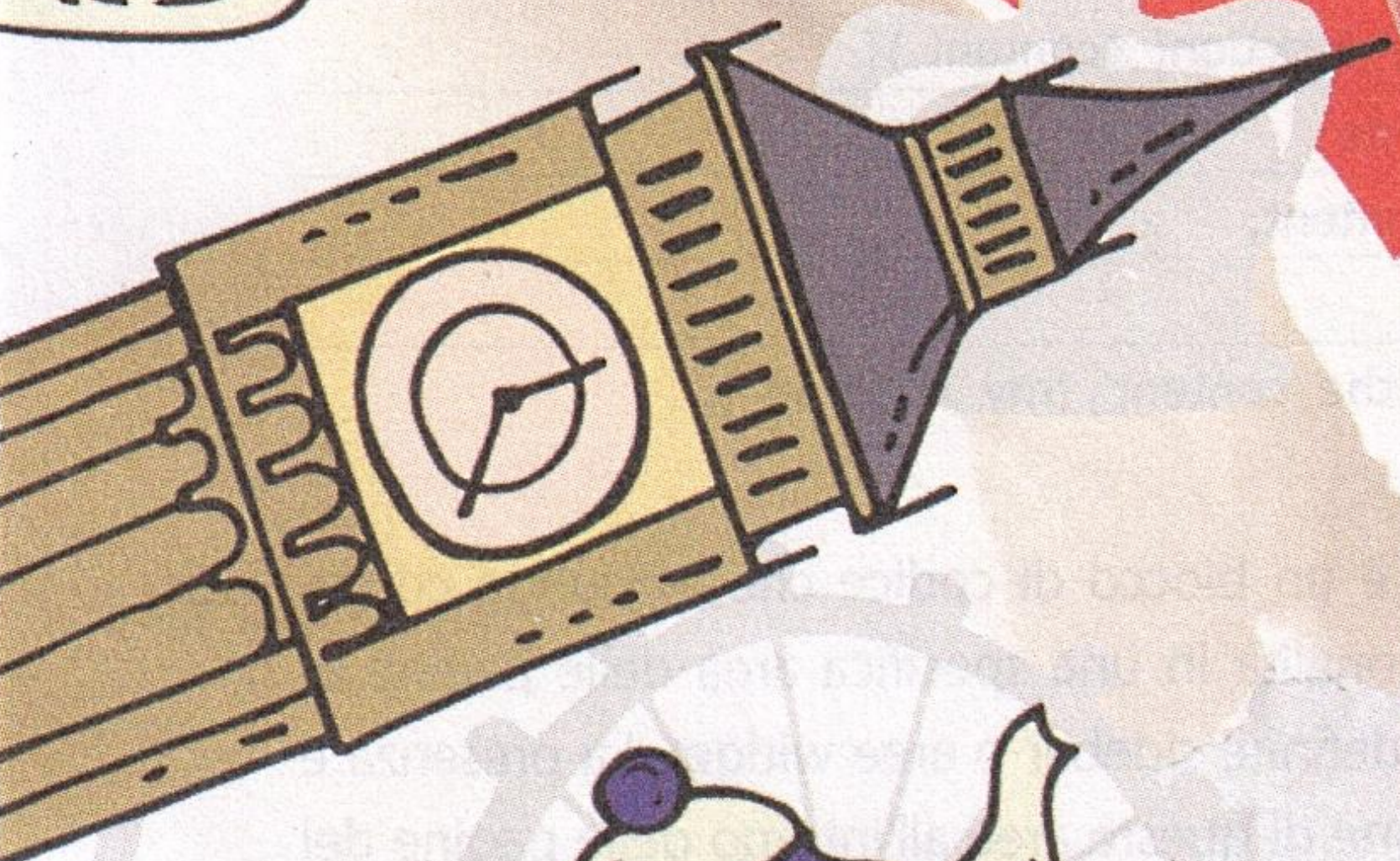
```
categories:
- conversazioni
conversations:
- - Buongiorno, come stai?
  - Sto bene, e tu?
  - Anche io sto bene.
  - Sono felice di sentirtelo dire.
  - Grazie.
- - La pizza è la cosa più buona del mondo.
  - La torta alla nutella è migliore della pizza.
  - Cos'altro è delizioso?
  - Niente, non credo esista qualche cosa di migliore
  - Dimmi come sei.
  - Che cosa vuoi sapere?
  - Sei un robot?
  - Si.
  - Cosa si prova?
  - Cos'è che vuoi sapere?
  - Come funzioni?
  - La domanda è molto complessa, hai una domanda
                                          di riserva?
  - No, sono molto stanco.
```

Anche in questo caso si possono inserire più conversazioni, iniziandone sempre una nuova con il simbolo - - . Le conversazioni possono essere anche piuttosto lunghe e in teoria dovrebbero svilupparsi come quelle che normalmente si sviluppano tra esseri umani. Le conversazioni che forniamo al bot per l'allenamento determineranno anche la personalità del bot stesso: tenderà a dare risposte più positive o più negative, più simpatiche o più ciniche, a seconda della conversazioni che gli sottoponiamo. Volendo è addirittura possibile fornire al bot, formattandole nel modo che abbiamo appena visto, delle conversazioni che abbiamo davvero avuto personalmente con un amico: se forniamo al bot una buona quantità di testi scritti dal nostro amico, vedremo che il bot molto spesso risponderà proprio come avrebbe fatto l'amico in questione. Questo perché il machine learning, in questo caso, non richiede nemmeno troppo tempo per apprendere il modo corretto di rispondere ai messaggi. Sicuramente, è aiutato dal fatto che le persone, in realtà, sono in genere molto più prevedibili di quello che si possa pensare, e in molti casi una macchina può davvero simulare il comportamento di un essere umano.

Luca Tringali

L'AUTORE

Luca Tringali ha studiato chimica presso l'università di Trieste, ma coltiva da più di quindici anni la passione per l'informatica. Collabora allo sviluppo di Kubuntu GNU/Linux ed al progetto KDE, utilizzando diversi linguaggi di programmazione. Si interessa soprattutto di intelligenza artificiale e domotica, oltre alla sicurezza informatica. Può essere contattato tramite l'indirizzo tringalinvent@libero.it

Fig.6: Dal terminale di Python possiamo vedere come proceda l'allenamento e il riconoscimento delle frasi e degli eventuali comandi.

UN PLUGIN PER WORDPRESS CON LE GOOGLE API

GRAZIE ALLE API DI WORDPRESS È POSSIBILE AGGIUNGERE FUNZIONALITÀ ETEROGENEE E CONTENUTI ESTERNI AI SITI. VEDIAMO UN MASH-UP TRA LE API DI WORDPRESS E LA GOOGLE CALENDAR API CHE SINCRONIZZA AUTOMATICAMENTE LE PAGINE DEL SITO CON IL NOSTRO CALENDARIO GOOGLE

Con oltre cinquantamila plugin ospitati gratuitamente, la Directory dei Plugin è un enorme archivio di software cui gli utenti di WordPress possono attingere per aggiungere (quasi) ogni tipo di funzionalità alle proprie istallazioni.
Grazie al lavoro di migliaia di sviluppatori, anche utenti non programmatori possono mettere in opera siti professionali, risparmiando denaro e dedicando le proprie risorse esclusivamente alla creazione e pubblicazione dei contenuti. Se questo è vero anche per siti di complessità media, può tuttavia capitare di non trovare un plugin che offra la specifica funzionalità di cui si ha bisogno, oppure il plugin esistente ne offre una versione limitata. E se anche una ricerca in rete tra i plugin commerciali rimane infruttuosa, non c'è altra soluzione che mettere mano al codice e cominciare a sviluppare il nostro primo plugin.

PRESENTAZIONE DEL PLUGIN

L'occasione ci viene offerta dalla necessità di inserire, nella sidebar di un qualsiasi tema, un widget che visualizzi un elenco di eventi registrati in un calendario pubblico di Google. Il problema che si pone è, quindi, quello di recuperare un certo numero di eventi da un calendario e visualizzarli sotto forma di elenco all'interno di un widget. A questo scopo ci serviremo di un plugin sviluppato facendo uso della Widgets API di WordPress e della Google Calendar API V3.Niente paura, però. Non sarà necessario entrare in profondità nell'API di Google, né tantomeno saranno indispensabili conoscenze avanzate di programmazione. Grazie alle potenzialità delle API di WordPress, sviluppare un buon plugin è un obiettivo alla portata anche di programmatori alle prime armi. Tutto ciò che necessita è una conoscenza base di programmazione ad oggetti in PHP. Prima di iniziare, consigliamo di scaricare il plugin che andiamo a descrivere dalla Directory dei Plugin, all'indirizzo https://wordpress.org/plugins/gcal-events-list/.

Conoscenze richieste
Buona conoscenza di WordPress.

Software
Web server con PHP 7 e MySQL 5.6 o MariaDB 10.0. In locale è consigliato il pacchetto XAMPP, scaricabile all'indirizzo http://www.apachefriends.org/it/xampp.html

Impegno
Tempo di realizzazione
DEFINIZIONI

Un plugin è un pacchetto software che estende le funzionalità core di un'istallazione di WordPress, permettendo di adattare il CMS ad ogni esigenza di pubblicazione: da piattaforma di e-Commerce a social network, da portfolio prodotti a Web-app interattiva. Per facilitare lo sviluppo dei plugin e rendere snello, sicuro ed efficiente il codice, WordPress fornisce la Plugin API (https://codex.wordpress.org/Plugin_API), ampiamente documentata nel Plugin Handbook (https://developer.wordpress.org/plugins/). Si tratta di un set di funzioni che vengono definite "hook", raggruppate nelle due categorie delle azioni e dei filtri.
Le azioni sono particolari funzioni che permettono di eseguire una funzione di callback ad un determinato momento dell'esecuzione del core. Queste vanno utilizzate come segue:

```
function my_function() {
  // your code
}
add_action( 'init', 'my_function' );
```

Nello slang degli sviluppatori, si dice che la funzione *my_function* viene "agganciata" all'hook *init*, il quale viene eseguito dopo il completo caricamento di WordPress, ma prima dell'invio di qualsiasi header. I filtri sono funzioni che permettono di modificare i dati restituiti da una diversa funzione. Per utilizzare i filtri è necessario definire una funzione di callback, la quale dovrà restituire in output il valore modificato della variabile passata in input. Ecco un esempio:

```
function my_function( $content ){
  // your code
  return $content;
}
add_filter( 'the_content', 'my_function' );
```

Un widget è un blocco di codice che aggiunge contenuti o funzionalità in una specifica area delle pagine di WordPress, definite sidebar o aree widget. La presenza e la collocazione di queste aree all'interno delle pagine del front-end dipende dal tema installato e potrebbe anche non essere supportata (maggiori informazioni al riguardo nel Theme Handbook https://developer.wordpress.org/

themes/functionality/widgets/). Se il tema supporta i widget, questi saranno gestibili nella pagina *Aspetto → Widget del pannello di amministrazione*. I widget vengono aggiunti alle installazioni tramite un plugin e un set di funzioni specifiche raggruppate nella Widgets API (https://codex.wordpress.org/Widgets_API).

LA STRUTTURA DI BASE DI UN PLUGIN DI WORDPRESS

Un plugin di WordPress è composto di almeno un file *.php*. Se il plugin è destinato ad essere pubblicato nella Directory di WordPress.org, sarà necessario un file *readme.txt*, al cui interno andranno inseriti dati e informazioni che saranno visualizzate nella pagina di download.

Il file *.php* dovrà avere un nome univoco. Ciò significa che non dovrà esistere un plugin con lo stesso nome nella Directory. Ciò in quanto, anche se questo non fosse destinato alla pubblicazione, potrebbero generarsi conflitti con plugin presenti nella Directory che potrebbero danneggiare l'intera istallazione.

Prima di assegnare il nome al file e decidere di conseguenza il nome del plugin, è quindi necessario fare una ricerca nella Directory.

Dopo questa avvertenza sul nome da assegnare al plugin, è necessaria una nota sullo stile del codice che si andrà a sviluppare. La community di sviluppatori di WordPress ha raggruppato le best practice di sviluppo in quattro diversi standard di programmazione:

- CSS Coding Standard
- HTML Coding Standard
- JavaScript Coding Standard
- PHP Coding Standard

Per gli scopi di questo articolo, si farà riferimento agli standard HTML e PHP. Maggiori informazioni sono disponibili nel manuale online di WordPress.org https://make.wordpress.org/core/handbook/best-practices/coding-standards/

L'INTESTAZIONE DEL PLUGIN

Un plugin destinato alla distribuzione nella Directory deve essere accompagnato da una serie di informazioni che vanno inserite, sotto forma di commento, nell'intestazione dello script principale.

Quello che segue è l'elenco dei dati ammessi:

Plugin Name: (richiesto) è il nome del plugin
Plugin URI: è la URL della risorsa che offre maggiori informazioni sul plugin (può anche essere la pagina dedicata su WordPress.org)
Description: è la descrizione del plugin (meno di 140 caratteri)
Version: è la versione corrente
Author: uno o più nomi separati da virgole (i nomi degli autori devono essere i nickname scelti per WordPress.org)
Author URI: la home page dell'autore del plugin
License: la licenza del plugin (ad esempio GPL2)

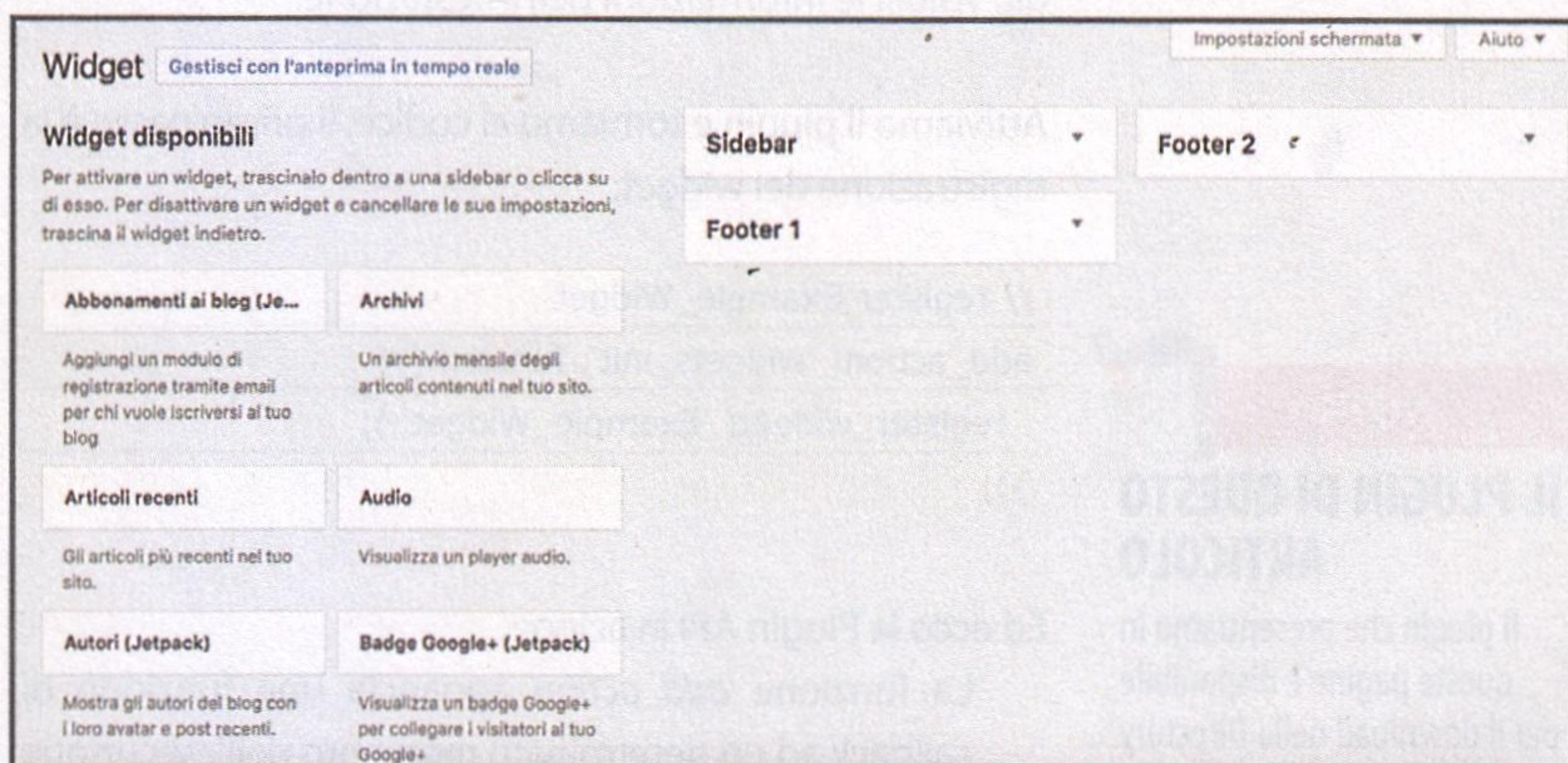



Fig. 1: La pagina dei widget del tema Twenty Seventeen

License URI: la URL dove trovare i dettagli della licenza (ad esempio https://www.gnu.org/licenses/gpl-2.0.html)
Text Domain: è il text domain del plugin
Domain Path: individua il percorso dei file delle traduzioni (si legga al riguardo https://developer.wordpress.org/plugins/internationalization/how-to-internationalize-your-plugin/)

Il plugin che accompagna questo articolo è fornito della seguente intestazione:

```
<?php
/**
 * @package Example Plugin
 * @version 0.1
 */
/*
Plugin Name: Example Plugin
Plugin URI: http://example.com/example-plugin/
Description: An example plugin showing a widget
Author: nickname
Version: 0.1
Author URI: http://yourwebsite.it/en/
License: GPLv2 or later
Text Domain: example-plugin
Network: true
*/
```

Maggiori informazioni sugli header sono disponibili all'indirizzo https://developer.wordpress.org/plugins/the-basics/header-requirements/.

REQUIREMENTS

WordPress funziona in qualunque ambiente che supporti PHP 7+ e MySQL 5.6+ o MariaDB 10.0+. Nonostante la flessibilità del software, è consigliato un server Apache o Nginx. Per gli utenti di Apache è opportuna l'abilitazione del mod_rewrite. Maggiori informazioni sui requisiti software sono disponibili all'indirizzo http://wordpress.org/about/requirements/

SVILUPPIAMO IL WIDGET

Una volta creato il file *example-plugin.php*, lo si salvi in una cartella avente lo stesso nome del file all'interno della directory */wp-content/plugins/*:

```
/wp-content/plugins/example-plugin/
```

Da questo momento, il plugin sarà visibile nella pagina dei plugin dell'istallazione di WordPress. Naturalmente, il plugin non effettuerà alcuna operazione, tuttavia saranno già visibili le informazioni dell'intestazione.

Attiviamo il plugin e torniamo al codice. Il primo passo è la registrazione del widget:

```
// register Example_Widget
add_action( 'widgets_init', function(){
  register_widget( 'Example_Widget' );
});
```

Ed ecco la Plugin API in azione.

- La funzione *add_action* aggancia una funzione di callback ad un determinato momento dell'esecuzione del core. In questo caso, si tratta di una funzione anonima che invoca, a sua volta, la funzione *register_widget*.
- *widget_init* è l'action hook che determina il momento di esecuzione della funzione di callback. *widget_init* viene eseguito non appena WordPress abbia registrato i widget predefiniti e individua il momento adatto alla registrazione di un nuovo widget.
- *register_widget* registra il widget individuato dall'argomento specificato.

Per creare un nuovo widget è necessario estendere la classe *WP_Widget* e ridefinire alcuni dei suoi metodi. Questa la struttura generale del codice:

```
class Example_Widget extends WP_Widget {
  // class constructor
  public function __construct() {}

  // output the widget content on the front-end
  public function widget( $args, $instance ) {}

  // output the option form field in admin Widgets screen
  public function form( $instance ) {}

```

Fig. 2: Prima di assegnare il nome al file e al plugin è necessario accertarsi che nella directory non esistano plugin omonimi

NOTA

IL PLUGIN DI QUESTO ARTICOLO

Il plugin che presentiamo in queste pagine è disponibile per il download nella Directory dei Plugin all'indirizzo https://wordpress.org/plugins/gcal-events-list/

```
  // save options
  public function update( $new_instance,
                          $old_instance ) {}
}
```

Analizziamo i metodi:

- *__construct()* è il costruttore di classe grazie al quale possono essere impostati alcuni parametri del widget;
- *widget()* genera e stampa il contenuto HTML che sarà visualizzato nel front-end del sito;
- *form()* genera gli elementi del form di configurazione del widget;
- *update()* memorizza i dati acquisiti dall'amministratore in fase di configurazione.

IL COSTRUTTORE DI CLASSE

Il costruttore di classe registra l'ID e il titolo del widget, nonché una serie di parametri, come nome della classe e descrizione:

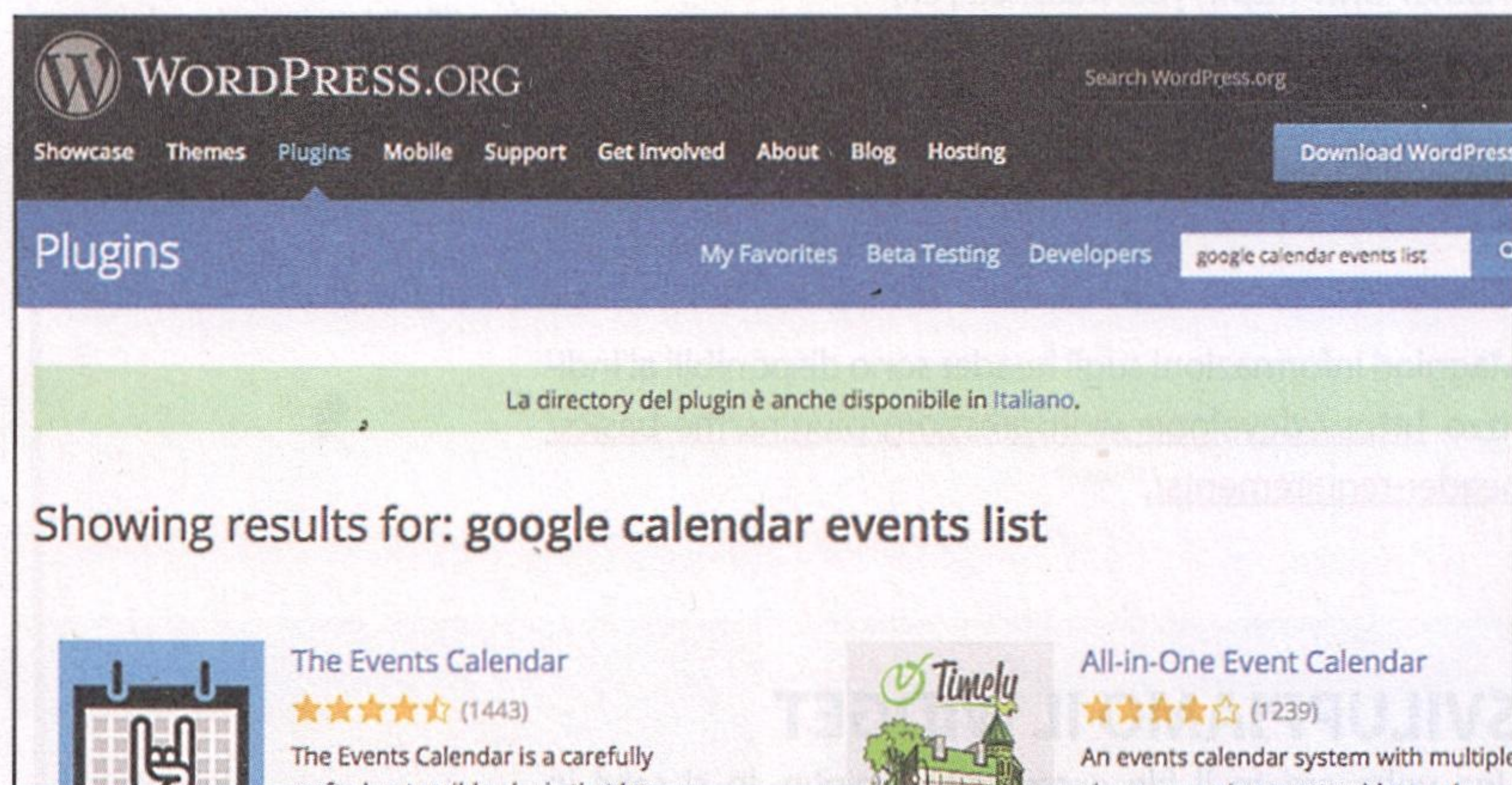
```
/**
 * Set up the widget name and options
 */
public function __construct(){

  $widget_ops = array( 'description' => 'An example
     widget that generates a list of events from a public
                                  Google Calendar' );

  parent::__construct( 'example-widget', 'Example
Widget', $widget_ops );

}
```

- *example-widget* è l'ID base del widget
- *'Example Widget'* è il titolo
- *$widget_ops* è un array di opzioni

Per l'elenco completo dei parametri di configurazione, si faccia riferimento alla documentazione online all'indirizzo https://developer.wordpress.org/reference/classes/wp_widget/__construct/.

IL FORM DI CONFIGURAZIONE

Dal lato admin, il widget sarà configurato e pubblicato nella pagina *Aspetto → Widget*. In questa pagina l'amministratore potrà aggiungere il widget alla sidebar prescelta, e quindi impostare i parametri di configurazione tramite un form. Il form viene generato dal metodo *form()* della classe *WP_Widget*. Analizziamo il primo frammento di codice.

```
public function form( $instance ){
```

GCal Events List
Disattiva | Modifica

GCal Events List generates a list of events from a public Google Calendar. You need the calendar ID to make it work.
Versione 2.1 | Di veleno | Visualizza i dettagli

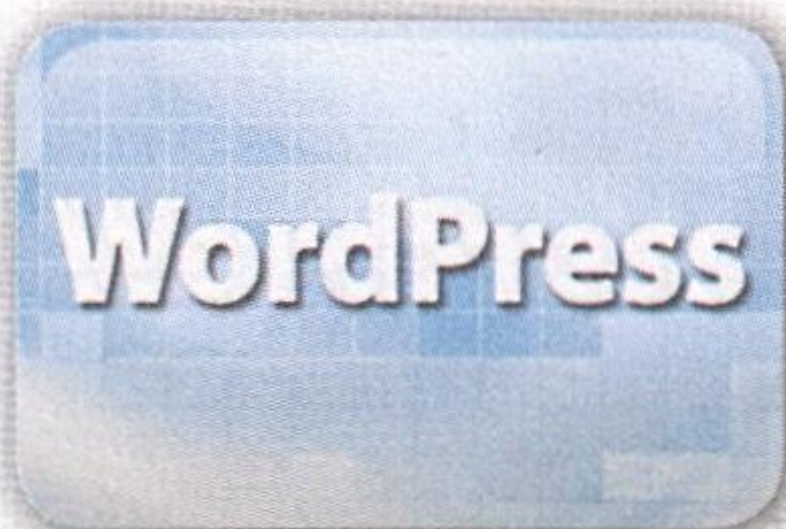



Fig.3: Una volta attivato, il plugin comparirà nell'elenco dei plugin dell'istallazione

```
$title = ! empty( $instance['title'] ) ? $instance['title'] :
                                  esc_html__( 'Title' );
$calendar = ! empty( $instance['calendar'] ) ?
                               $instance['calendar'] : '';
$key = ! empty( $instance['key'] ) ? $instance['key']
                                                     : '';
$timeMin = ( ! empty( $instance['timeMin'] )
            && isDate( $instance['timeMin'] ) ) ?
          $instance['timeMin'] : date( 'Y-m-d' );
$maxResults = ! empty( $instance['maxResults'] ) ?
                             $instance['maxResults'] : '5';
```

Il metodo *form()* accetta come argomento un array che memorizza le impostazioni del widget. Le impostazioni del nostro widget sono le seguenti:

title: (*string*) è il titolo del widget;
calendar: (*string*) è l'ID del calendario pubblico cui si intende collegare il widget (visibile nelle impostazioni del calendario di Google);
key: (*string*) è la chiave API fornita da Google (ottenibile all'indirizzo https://developers.google.com/apis-explorer/);
timeMin: (*datetime*) è il limite minimo espresso in formato timestamp RFC3339, con indicazione obbligatoria della timezone (per maggiori informazioni si veda la documentazione di Google (https://developers.google.com/google-apps/calendar/v3/reference/events/list);
maxResults: è il numero massimo di eventi da visualizzare;

Gli operatori ternari stabiliscono valori di default nel caso in cui non esistano valori precedentemente memorizzati in archivio. Proseguiamo aggiungendo il seguente codice:

```
?>
<p>
 <label for="<?php echo $this->get_field_id( 'title' );
                                              ?>">Title:
  <input class="widefat"
  id="<?php echo $this->get_field_id( 'title' ); ?>"
  name="<?php echo $this->get_field_name( 'title' );
                                                     ?>"
  type="text"
  value="<?php echo $title; ?>" />
 </label>
</p>
```

Il codice qui sopra genera il primo campo del form di configurazione del widget. I valori degli attributi degli elementi label e input vengono impostati grazie ai metodi *get_field_id* e *get_field_name*. In modo simile creiamo un campo in cui inserire l'ID del Google Calendar da collegare al widget ed un campo per la chiave API di Google:

```
<p>
 <label for="<?php echo $this->get_field_id( 'calendar'
                                 ); ?>">Calendar ID:
  <input class="widefat"
  id="<?php echo $this->get_field_id( 'calendar' ); ?>"
  name="<?php echo $this->get_field_name
                                  ( 'calendar' ); ?>"
  type="text"
  value="<?php echo $calendar; ?>" />
 </label>
</p>
<p>
 <label for="<?php echo $this->get_field_id( 'key' );
                                   ?>">Your API key:
  <input class="widefat"
  id="<?php echo $this->get_field_id( 'key' ); ?>"
  name="<?php echo $this->get_field_name( 'key' );
                                                 ?>"
  type="text"
  value="<?php echo $key; ?>" />
 </label>
</p>
```

Creiamo, poi, un campo in cui inserire la data minima degli eventi da visualizzare:

```
<p>
 <label for="<?php echo $this->get_field_id('timeMin');
?>">Start from (YYYY-MM-DD):
  <input class="widefat"
  id="<?php echo $this->get_field_id('timeMin'); ?>"
  name="<?php echo $this->get_field_
name('timeMin'); ?>"
  type="text"
  value="<?php echo $timeMin; ?>" />
 </label>
</p>
```

L'etichetta del campo suggerisce all'utente quale sia il formato data da utilizzare (*YYYY-MM-DD*).
Il campo successivo è leggermente più complesso, trattandosi di un menu di selezione:

```
<p>
```

NOTA

COSA SONO I WIDGET

Un widget è un blocco di codice che aggiunge contenuti o funzionalità in una specifica area delle pagine di WordPress, definite sidebar o aree widget. Questi vengono aggiunti alle installazioni tramite un plugin e un set di funzioni specifiche raggruppate nella Widgets API (https://codex.wordpress.org/Widgets_API).

NOTA

ACTION HOOK E FILTER HOOK

Si tratta di funzioni particolari attivate in specifici momenti dell'esecuzione del core, a cui è possibile "agganciare" funzioni di callback definite dall'utente. WordPress fornisce due tipi di hook: i filtri, che permettono di modificare l'output di una funzione prima che questo venga inviato al client, e le azioni, che sono paragonabili agli eventi JavaScript: una volta agganciata una funzione di callback, questa viene eseguita automaticamente all'attivazione dell'azione. Nel Codex la documentazione completa: https://codex.wordpress.org/Plugin_API/Hooks

```
<label for="<?php echo $this->get_field_
    id('maxResults'); ?>"><?php echo __('Max results');
                                          ?>: </label>
<select id="<?php echo $this->get_field_id( 'max
    Results' ); ?>" name="<?php echo $this->get_field_
        name( 'maxResults' ); ?>" class="widefat">
  <option <?php if ( $maxResults == '1' ) echo
        'selected="selected"'; ?>>1</option>
  <option <?php if ( $maxResults == '3' ) echo
        'selected="selected"'; ?>>3</option>
  <option <?php if ( $maxResults == '5' ) echo
        'selected="selected"'; ?>>5</option>
  <option <?php if ( $maxResults == '10' ) echo
        'selected="selected"'; ?>>10</option>
  <option <?php if ( $maxResults == '15' ) echo
        'selected="selected"'; ?>>15</option>
  <option <?php if ( $maxResults == '20' ) echo
        'selected="selected"'; ?>>20</option>
</select>
</p>
<?php
}
```

Gli elementi *option* stabiliscono i valori ammessi. Il risultato del lavoro è visibile nella figura 3.

IL SALVATAGGIO DEI DATI

Il metodo *update()* registra i dati inseriti nel modulo di configurazione:

```
public function update( $new_instance, $old_instance ){
```

Fig.4: Il modulo di configurazione del widget

```
$instance = array();

$instance['title'] = ( ! empty( $new_instance['title'] ) )
    ? strip_tags( $new_instance['title'] ) : '';

$instance['calendar'] = ( ! empty( $new_
    instance['calendar'] ) ) ? strip_tags( $new_
        instance['calendar'] ) : '';

$instance['key'] = ( ! empty( $new_instance['key'] ) )
    ? strip_tags( $new_instance['key'] ) : '';

$instance['timeMin'] = ( ! empty( $new_
    instance['timeMin'] ) && isDate( $new_
    instance['timeMin'] ) ) ? strip_tags( $new_
    instance['timeMin'] ) : date( 'Y-m-d' );

$instance['maxResults'] = ( ! empty(
    $new_instance['maxResults'] ) ) ? strip_tags( $new_
        instance['maxResults'] ) : '5';

return $instance;
}
```

update() accetta come argomenti due array di parametri: il primo contiene i valori trasmessi dall'utente (*$new_instance*), il secondo i valori presenti in archivio (*$old_instance*). Grazie agli operatori ternari, vengono verificati i dati trasmessi e assegnati i valori agli elementi dell'array *$instance*, che viene quindi restituito in uscita dalla funzione. Nessun'altra operazione è necessaria per il salvataggio dei dati, che sarà operato direttamente da WordPress.

IL WIDGET NEL FRONT-END

Il metodo *widget()* produce il codice HTML che sarà utilizzato da WordPress per la visualizzazione nel front-end.

```
public function widget( $args, $instance ){

  echo $args['before_widget'];
  //widget title
  if ( ! empty( $instance['title'] ) ) {
    echo $args['before_title'] . apply_filters( 'widget_
        title', $instance['title'] ) . $args['after_title'];
  }

  //google calendar parameters
  $params = array(
    'calendar' => ! empty( $instance['calendar'] ) ?
            $instance['calendar'] : '', //calendar ID
    'key' => ! empty( $instance['key'] ) ?
            $instance['key'] : '',
    'timeMin' => ( ! empty( $instance['timeMin'] ) &&
  isDate( $instance['timeMin'] ) ) ? $instance['timeMin'] :
            date( 'Y-m-d' ),
```

```
    'maxResults' => ! empty( $instance['maxResults'] ) ?
                          $instance['maxResults'] : '5',
  );

  if( ! empty( $params['calendar'] ) ){
    gcal_getData( $params );
  }else{
    echo __( 'You shoud set the calendar ID to make this
                                   widget work' );
  }

  //close div
  echo $args['after_widget'];
}
```

widget() accetta come argomenti due array: *$args* è un array di parametri che determinano il comportamento generale dei widget di una determinata sidebar (si veda il Codex per una descrizione più completa https://codex.wordpress.org/Function_Reference/register_sidebar); *$instance* è l'array delle impostazioni del widget già descritto in precedenza.

L'array *$parames*, definito all'interno della funzione, memorizza i valori dei parametri *calendar*, *key*, *timeMin* e *maxResults* che saranno utilizzati successivamente per interrogare l'API di Google.

La funzione custom *isDate()* permette di verificare che la stringa trasmessa come argomento rispetti il formato data *Y-m-d* (maggiori info sulla classe PHP DateTime all'indirizzo http://php.net/manual/en/class.datetime.php). La funzione è definita come segue:

```
function isDate( $date ){

  $dateTime = new DateTime();

  $d = $dateTime->createFromFormat( 'Y-m-d', $date );

  return $d && $d->format( 'Y-m-d' ) === $date;
}
```

Torniamo al metodo *widget()*. Una volta definito l'array e assegnati i valori degli elementi, viene invocata la funzione custom *gcal_getData()*, che ha lo scopo di prelevare i dati dal calendario specificato. Vediamola in dettaglio:

```
function gcal_getData( $params ){

  $calendarID = urlencode( $params['calendar'] );
  $key = $params['key'];
  $maxResults = ! empty( $params['maxResults'] ) ?
      ( "&maxResults=" . $params['maxResults'] ) : '5';

  if( ! empty( $params['timeMin'] ) && isDate(
                                   $params['timeMin'] ) ){

    $RFC3339 = $params['timeMin'] . 'T00:00:00Z';
```

FESTIVITÀ ITALIANE

Assunzione / Ferragosto
2017-08-15

Tutti i Santi
2017-11-01

Immacolata
2017-12-08

Natale
2017-12-25

Santo Stefano
2017-12-26

Fig.5: Il widget del Google Calendar nel front-end del sito

```
    $timeMin = '&timeMin=' . $RFC3339;

  }else{

    $timeMin = '&timeMin=' . date( 'Y-m-d' ) .
                                         'T00:00:00Z';

  }

  $resource = "https://www.googleapis.com/
  calendar/v3/calendars/" . $calendarID . "/events?key="
           . $key . $maxResults . "&orderBy=startTime" .
                         "&singleEvents=true" . $timeMin;

  $request = wp_remote_get( $resource );

  if( is_wp_error( $request ) ) {
    return false;
  }

  $body = wp_remote_retrieve_body( $request );
  $data = json_decode( $body );

  if( ! empty( $data->items ) ) {
  ?>
    <ul>
      <?php foreach( $data->items as $item ) { ?>
      <li><?php echo $item->summary; ?><br />
      <?php echo $item->start->date; ?>
      </li>
```

IL CODEX E GLI HANDBOOK

Il Codex è il riferimento ufficiale di WordPress, dove gli sviluppatori e gli utenti possono trovare manuali, articoli, specifiche tecniche ed esempi concreti di tutto quello che riguarda WordPress. Tutto quanto abbiamo descritto nell'articolo è ampiamente documentato nel Codex, a cui facciamo ampio rinvio per i necessari approfondimenti (https://codex.wordpress.org/)

Oltre al Codex, WordPress.org fornisce due manuali estremamente utili sia per i più esperti, che per gli sviluppatori alle prime armi: il Plugin Handbook (https://developer.wordpress.org/plugins/) e il Theme Handbook (https://developer.wordpress.org/themes/).

```
<?php } ?>
</ul>
<?php
}else{
  echo __( 'No items found!' );
}
}
```

Il compito della funzione è quello di utilizzare i parametri acquisiti dall'amministratore del sito per generare una URL da trasmettere all'API di Google. La URL dovrà avere la seguente struttura:

https://www.googleapis.com/calendar/v3/calendars/calendarId/events

La URL presenta un primo parametro interno (*path parameter*) che rappresenta l'ID del calendario. Alla URL andranno aggiunti i parametri della query string: *key, maxResults, orderBy, singleEvents, timeMin*. Maggiori informazioni all'indirizzo https://developers.google.com/google-apps/calendar/v3/reference/events/list.
Il prelievo e la decodifica dei dati avviene attraverso le funzioni della HTTP API di WordPress (https://codex.wordpress.org/HTTP_API):

```
$request = wp_remote_get( $resource );

if( is_wp_error( $request ) ) {
  return false;
}

$body = wp_remote_retrieve_body( $request );
$data = json_decode( $body );
```

La funzione *wp_remote_get()* recupera i dati grezzi di una risposta *HTTP* utilizzando il metodo *GET*. La funzione accetta come argomenti la URL della richiesta ed un array di parametri facoltativi, qui non presente, e restituisce la risposta del servizio o un oggetto *WP_Error* in caso di errore. La condizione permette di verificare se il server abbia risposto correttamente.
wp_remote_retrieve_body() recupera il *body* della risposta in formato JSON, che viene quindi passato alla funzione *PHP json_decode()* per la decodifica.
Il successivo ciclo *foreach* itera tra gli elementi dell'array di dati per generare il contenuto del widget:

```
if( ! empty( $data->items ) ) {
?>
  <ul>
    <?php foreach( $data->items as $item ) { ?>
    <li><?php echo $item->summary; ?><br />
    <?php echo $item->start->date; ?>
    </li>
  <?php } ?>
  </ul>
<?php
}else{
  echo __( 'No items found!' );
}
```

Il plugin è ora perfettamente funzionante, ma non è ancora pronto per la pubblicazione.

PERCHÉ PUBBLICARE UN PLUGIN NELLA DIRECTORY

I motivi che possono consigliare di distribuire un plugin attraverso la Directory di WordPress.org sono diversi. Il primo di questi motivi è quello di guadagnare visibilità come sviluppatore. Le pagine dei plugin sono indicizzate ottimamente da Google e questo implica una grande visibilità per il lavoro di uno sviluppatore. Tale visibilità si trasforma in reputazione nel caso in cui il plugin riscuota un buon successo di pubblico.
Inoltre, la Directory fornisce gratuitamente una serie di servizi professionali, come una pagina dedicata con immagini, descrizione, statistiche, changelog, forum di supporto, ed altro ancora. Chi decida di distribuire le proprie produzioni non potrà fare a meno di considerare attentamente questa opportunità.
Altro motivo da considerare è meno utilitaristico, ma altrettanto importante: contribuire con il proprio lavoro alla crescita della community che gravita intorno a WordPress. In effetti, WordPress è un software gratuito, come sono gratuiti la maggior parte dei temi e dei plugin disponibili. Mettere a disposizione della community i frutti del proprio lavoro è il modo migliore per ringraziare gli sviluppatori e gli utenti che contribuiscono gratuitamente, da ogni parte del mondo, alla crescita del CMS.

A COSA DOBBIAMO FARE ATTENZIONE

Si tenga presente che se la pubblicazione nella Directory è un'opportunità per promuovere la propria immagine di sviluppatori, dall'altro lato può essere un rischio se non si rispettano alcune regole fondamentali di programmazione e di comportamento.
Si è già detto che, per chi sviluppa temi e plugin per WordPress, è fortemente raccomandato seguire le linee guida previste dalla documentazione online. A questo proposito, è consigliata una preventiva lettura delle *Plugin Guidelines* (https://developer.wordpress.org/plugins/wordpress-org/detailed-plugin-guidelines/).
Per quanto riguarda la programmazione, è buona regola attenersi alla seguente serie di raccomandazioni.

Sviluppiamo codice *human readable*. Ciò consente a chiunque di comprendere il codice del plugin, e possibilmente di inviare feedback che possono aiutare ad eliminare eventuali bugs e suggerire modifiche e integrazioni. Anche per questo è importante ade-

NOTA

IL CODEX

Il Codex è il riferimento ufficiale di WordPress, dove gli sviluppatori e gli utenti possono trovare manuali, articoli, specifiche tecniche ed esempi concreti di tutto quello che riguarda WordPress.
Tutto quanto abbiamo descritto in articolo è ampiamente documentato nel Codex, a cui facciamo ampio rinvio per i necessari approfondimenti. https://codex.wordpress.org/

guare il codice ai *WordPress Coding Standard* (https://make.wordpress.org/core/handbook/best-practices/coding-standards/), le linee guida cui devono attenersi gli sviluppatori di WordPress. Commentiamo sempre il nostro codice, al fine di rendere immediatamente chiaro quale task compie una determinata funzione, quali argomenti accetta, quale valore restituisce. Nei commenti possono essere aggiunte le URL a risorse esterne e ogni altro tipo di informazione che possa essere utile non solo ad altri sviluppatori, ma anche a noi stessi, soprattutto quando il codice è lungo e complesso, oppure se ci si ritorna sopra dopo lungo tempo. É fondamentale, a questo riguardo, un'attenta lettura degli *Inline Documentation Standards* (https://make.wordpress.org/core/handbook/best-practices/inline-documentation-standards/). Utilizziamo sempre le funzioni di WordPress al posto delle corrispondenti funzioni PHP. Una funzione del framework non svolge mai lo stesso task della corrispondente funzione PHP, e ciò rende le prime più adatte allo sviluppo di applicazioni per WordPress, soprattutto sotto il profilo della sicurezza. Per una visione completa delle API di WordPress, si faccia riferimento alla documentazione online (https://codex.wordpress.org/WordPress_APIs).

Considerazioni di carattere più generale riguardano l'aspetto funzionale del software.

Anche se il plugin viene distribuito gratuitamente, va considerato come un prodotto da vendere, per il quale si ottiene reputazione invece che denaro. Per questo motivo è necessario assicurarsi che il plugin si faccia riconoscere dai plugin concorrenti, offrendo funzionalità uniche o, in alternativa, apportando novità sotto il profilo dell'accessibilità e dell'usabilità.
Stiamo sempre attenti alle richieste di supporto che ci provengono dagli utenti del plugin. Cerchiamo di fornire risposte chiare e tempestive e cerchiamo di aiutare gli utenti a risolvere eventuali problemi. Teniamo nella massima considerazione il feedback, e diamo valore ad ogni informazione che ci viene resa dalla community.
Aggiorniamo regolarmente il plugin, risolvendo errori ed eliminando bug e, quando il tempo ce lo permette, aggiungiamo nuove funzionalità.

LA LICENZA DI DISTRIBUZIONE

WordPress è rilasciato sotto licenza *GPLv2* (https://www.gnu.org/licenses/gpl-2.0.html) e tutti i prodotti derivati devono essere rilasciati sotto la stessa licenza o sotto una licenza compatibile, ciò anche se si progetta un plugin il cui uso preveda il pagamento di un canone. Per un elenco delle licenze compatibili, si legga *Various Licenses and Comments about Them* (https://www.gnu.org/licenses/license-list.en.html). Finalmente è il momento di preparare il plugin per la pubblicazione.

PREPARIAMO IL PLUGIN PER LA PUBBLICAZIONE

Per pubblicare un plugin nella Directory di WordPress.org, bisognerà dotarlo di un file *readme.txt*, di un header con i metadati richiesti, di una o più immagini (*assets*) da visualizzate nella pagina dedicata.
Degli header si è già parlato in precedenza. Qui analizziamo il contenuto del file *readme.txt*, il quale memorizza informazioni specifiche per la pubblicazione. Questi i campi disponibili:

Plugin name
Contributors (un elenco di user ID di WordPress.org)
Donate link
Tags (una o più etichette per favorire la ricerca)
Requires at least (versione di WordPress)
Tested up to (versione di WordPress)
Stable tag (versione del plugin)
License (ad es. GPLv2 or later)
License URI (ad es. https://www.gnu.org/licenses/gpl-2.0.html)
Description
Installation
Screenshots
Changelog

Le etichette sono abbastanza chiare da non richiedere descrizione. Un chiarimento solo per il campo *Screenshots*, dove va indicata la descrizione di ognuna delle immagini che accompagneranno il plugin. WordPress.org fornisce un template cui far riferimento nella creazione del file readme, disponibile all'indirizzo https://wordpress.org/plugins/files/2017/03/readme.txt. Per snellire il lavoro di compilazione, si può far ricorso al *Readme Generator* (https://generatewp.com/plugin-readme/) di GenerateWP.
Infine, un utile strumento di verifica della correttezza del file è il *Readme Validator* (https://wordpress.org/plugins/developers/readme-validator/) di WordPress.org.

LA STRUTTURA DEL PLUGIN

Prima di procedere alla pubblicazione è necessario avvertire che i file saranno archiviati nella repository SVN di WordPress.org. Da questo archivio chiunque avrà la possibilità di prelevare una copia dei file, ma solo l'autore avrà i privilegi per caricarli. Inoltre, una volta pubblicato il plugin, l'autore potrà modificare o eliminare i file esistenti, oppure aggiungerne di nuovi, come pure creare nuove versioni del plugin. Il tutto sarà registrato dalla subversion.
Nella repository ogni plugin viene strutturato nelle seguenti cartelle:

/assets/
/branches/
/tags/
/trunk/

Nella cartella */assets/* andranno collocati screenshots, immagini di testa (header) e icone; nella cartella */trunk/* andranno invece inseriti i file del plugin; branche divergenti andranno nella cartella */branches/*; le varie versioni andranno nella cartella */tags/*.

IMMAGINI E ICONE

Gli elementi grafici che compaiono nella pagina dedicata al plugin nella Directory di WordPress.org, collocati come già detto nella cartella */assets/*, si distinguono in banner, icone e screenshot.
Nomi, dimensioni e formati di queste immagini non sono arbitrarie, ma devono rispettare alcune regole. Sono, innanzi tutto, ammessi solo i formati PNG e JPG (per le icone è ammesso anche il formato SVG).
Le dimensioni sono predeterminate, come pure la nomenclatura dei file.
Per i banner si hanno a disposizione i seguenti nomi e formati:

banner-772x250.(jpg|png) per il formato regolare;
banner-1544x500.(jpg|png) per i display retina.

Le icone sono immagini quadrate con i seguenti nomi/dimensioni:

icon-128×128.(jpg|png)
icon-256×256.(jpg|png)
icon.svg

Infine, gli screenshot seguono il seguente schema:

screenshot-1.(jpg|png)
screenshot-2.(jpg|png)

Gli screenshot sono le immagini che compaiono nel corpo delle pagine dei plugin e, come si ricorderà, ricevono una descrizione nel file *readme.txt*.
Per questi non sono richieste dimensioni specifiche.

LA PUBBLICAZIONE DEL PLUGIN NELLA DIRECTORY

Prima di iniziare la pubblicazione, è necessario iscriversi a WordPress.org (https://login.wordpress.org/register) e inviare il plugin per la preventiva approvazione (https://wordpress.org/plugins/developers/add/). Il plugin sarà accodato per essere verificato manualmente dallo staff di WordPress.org (è possibile un'attesa di diversi giorni per ricevere l'approvazione, ma generalmente questa si riduce ad un paio di giorni). Se il plugin supera la verifica, si riceverà un'email di conferma; in caso negativo, il team evidenzierà eventuali problemi che dovranno essere risolti prima di una nuova verifica.
Ricevuto l'OK, si può dare avvio alle operazioni di pubblicazione. Il primo compito è il caricamento del plugin nella repository SVN. Si potrà utilizzare un client SVN, oppure semplicemente la command line del proprio sistema operativo.
Dunque, ricevuta l'email di approvazione, creiamo sul nostro HDD la directory che ospiterà la versione locale del plugin. Possiamo procedere con i consueti comandi del sistema, oppure tramite command line. Localizziamoci nella directory che dovrà ospitare i nostri progetti e creiamo il folder del progetto in corso:

```
$ mkdir localdir
```

Ora proviamo l'accesso alla repository:

```
$ svn co https://plugins.svn.wordpress.org/plugin-name
path/to/localdir
```

Otterremo la seguente risposta:

```
> A    plugin-name/branches
> A    plugin-name/tags
> A    plugin-name/trunk
> Checked out revision 99999999.
```

A indica che la directory corrispondente è stata copiata dalla repository nella directory locale (Added).
A questo punto possiamo aggiungere i file del plugin nella directory locale *plugin-name/trunk*, e quindi caricare i file

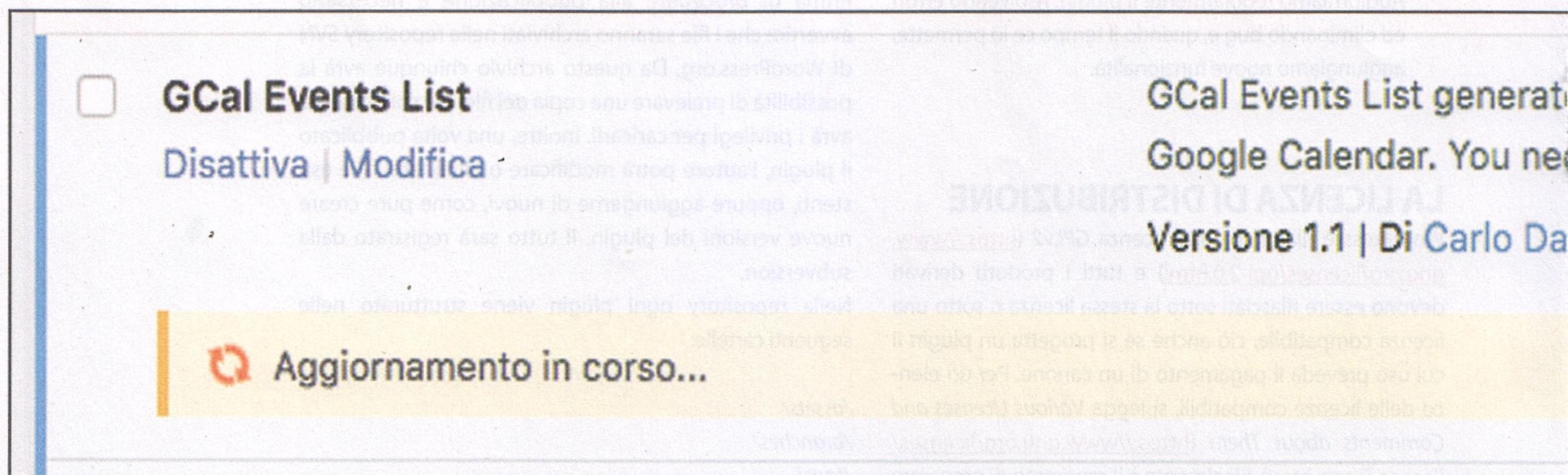



Fig. 6: Dopo la pubblicazione nella Directory, il plugin potrà essere aggiornato da tutti gli utenti senza lasciare il pannello di amministrazione di WordPress

GCal Events List
Disattiva | Modifica
GCal Events List generat
Google Calendar. You ne
Versione 1.1 | Di Carlo Da
Aggiornamento in corso...

sulla repository con il seguente comando:

```
localdir/$ svn add trunk/*
```

WordPress.org ci chiederà di autenticarci con le credenziali create in fase di registrazione. A operazioni concluse, si otterrà la seguente risposta:

```
> A    trunk/my-plugin.php
> A    trunk/readme.txt
```

É importante ricordare di non inserire i file del plugin in una sottodirectory di trunk, altrimenti il donload da WordPress.org non funzionerà correttamente.
A questo punto registriamo il tipo di modifica effettuata:

```
localdir/$ svn ci -m 'First plugin version'
```

La repository fornirà la seguente risposta:

```
> Adding trunk/my-plugin.php
> Adding trunk/readme.txt
> Transmitting file data .
> Committed revision 9999999.
```

Il passo successivo è l'inserimento dei file multimediali:

```
localdir/$ svn add assets/*
```

Registriamo ancora le modifiche effettuate:

```
localdir/$ svn ci -m 'Assets upload'
```

Il caricamento del plugin si conclude così. Ora è possibile accedere alla pagina del plugin su WordPress.org e provare il primo download.

IL RILASCIO DI UNA NUOVA VERSIONE

É importante aggiornare regolarmente i propri plugin. Nel tempo possono emergere errori, oppure potrebbero rendersi opportuni dei miglioramenti delle funzionalità esistenti, o aggiunte di nuove funzionalità. Quando è pronta la nuova versione è necessario, prima dell'upload, aggiornare il campo *Stable Tag* nel file *trunk/readme.txt* e il campo *Version number* nell'intestazione del plugin. A questo punto si procede a caricare i file della nuova versione in una sottocartella di *plugin-name/tag*, assegnando al nuovo folder il nome della versione:

```
localdir/$ svn cp trunk tags/1.0.1
> A tags/1.0.1
```

Facciamo, quindi, il check-in delle modifiche effettuate:

```
localdir/$ svn ci -m "tagging version 1.0.1"
> Adding      tags/1.0.1
> Adding      tags/1.0.1/my-plugin.php
> Adding      tags/1.0.1/readme.txt
> Committed revision 999999.
```

La nuova versione è online.

MODIFICHE MINORI

Saltuariamente potrebbero rendersi necessarie delle modifiche minori, che non richiedono il rilascio di una nuova versione, ma solo la correzione di un file già presente nella repository. Per prima cosa aggiorniamo la copia del plugin presente nel folder *trunk* locale:

```
localdir/$ svn up
> At revision 999999.
```

Modifichiamo il file e verifichiamo le variazioni:

```
localdir$ svn stat
> M       trunk/readme.txt
```

M indica che il file è stato modificato.
Vediamo, quindi, cosa è cambiato confrontando le due versioni:

```
localdir$ svn diff
```

Infine, registriamo le modifiche:

```
localdir$ svn ci -m "minor changes"
> Sending trunk/readme.txt
> Transmitting file data .
> Committed revision 999999.
```

Questo è tutto. Per eventuali approfondimenti, si legga il Plugin Developer Handbook di WordPress.org (https://developer.wordpress.org/plugins/).

Carlo Daniele

es a list of events from a public
d the calendar ID to make it work.
niele | Visualizza i dettagli

XAMARIN VS CORDOVA: QUALE SCEGLIERE?

LO SVILUPPO CROSS PLATFORM È ORMAI DIFFUSISSIMO A CAUSA DELLA SEMPRE MAGGIORE ETEROGENEITÀ DEI DEVICE. ESISTONO MOLTI STRUMENTI PER LO SVILUPPO MULTIPIATTAFORMA, CERCHEREMO DI CAPIRE VANTAGGI E SVANTAGGI DEI DUE PIÙ DIFFUSI: XAMARIN E CORDOVA

Sviluppare un'app per mobile, vuol dire creare molte versioni dello stesso progetto, ogni versione sviluppata con il codice nativo del produttore hardware. Semplificando, creare una app per iOS e per Android, vuol dire scrivere due volte lo stesso progetto software, in linguaggi diversi. Con Xamarin e Cordova, questo non è più necessario. Un unico framework permette di scrivere una volta sola la logica dell'applicazione.

Fino a una quindicina di anni fa, lanciare un nuovo prodotto software significava sostanzialmente scrivere un'applicazione Windows. Dieci anni fa, significava scrivere un sito Web che funzionava in Internet Explorer. Oggi, lanciare un prodotto di successo significa rilasciare versioni per Mac, Windows, iOS e Android. E potrebbe significare anche rilasciare versioni di tvOS, WatchOS, Tizen, XBOX, Windows Phone, forse anche altro. È incredibile, quante cose cose sono cambiate in 10 anni. Oggi, grazie ad Angular2 e Typescript, insieme a Cordova ed Electron è possibile abbassare la curva di apprendimento e i costi associati al *deploy* su più piattaforme. Sì! È vero: scrivere il codice una volta, eseguirlo ovunque è finalmente una realtà, con applicazioni che vengono eseguite in modo coerente ed efficiente. Alcuni anni fa, il *deploy* su piattaforme multiple significava scrivere codice per ogni piattaforma separatamente. In altre parole, dovevamo scrivere codice "nativo". L'argomento a favore del codice nativo è che fornisce prestazioni migliori e un migliore accesso alle funzioni avanzate delle rispettive piattaforme. Ma la scrittura di codice nativo ha due enormi svantaggi: una curva di apprendimento molto più elevata e i relativi costi associati, assieme alla necessità di riscrivere il codice per tutte le piattaforme su cui vogliamo giri la nostra app. In realtà è possibile avere il meglio dei due mondi: con Xamarin possiamo scrivere il codice in C# ed eseguire l'app ovunque come nativa. Xamarin, un progetto ormai interamente acquisito da Microsoft, è un'alternativa incredibile, che permette di utilizzare un solo linguaggio e un unico codice di base per effettuare il *deploy* delle applicazioni su piattaforme multiple. Allora perché non usare solo Xamarin, sempre? I programmatori "puristi" insistono sul fatto che se non si sta sviluppando al 100% con codice nativo, stai sviluppando male. C'è quasi una guerra di religione tra queste due scuole di pensiero. La realtà è che la maggior parte degli sviluppatori "smart" scelgono l'approccio e la piattaforma da adottare in base alle esigenze del singolo progetto. In questo articolo vedremo che è possibile costruire un'applicazione mobile utilizzando il meglio di ogni tecnologia. Xamarin e Cordova hanno ognuno i loro proseliti e talvolta il mischiare i due ambienti vi offre i migliori risultati possibili. In questo articolo faremo proprio questo, andremo ad analizzare una applicazione scritta sia in Cordova che in Xamarin. Questa applicazione di esempio ha un compito semplice: inviare dati a un server.

PUNTI DI FORZA E DI DEBOLEZZA

Per linguaggio nativo si intende *Swift* o *ObjectiveC* per *iOS* e *MacOS*, *Java* per *Android* e *C #* e *XAML* per *Windows*. Il vantaggio qui è il rendimento, i file binari più piccoli e il miglior supporto della piattaforma per il linguaggio. Lo svantaggio è quello di dover imparare tre diverse piattaforme, di mantenere tre diversi codici sorgente e di andare a effettuare per tre volte l'onerosa fase di debug. Questo approccio è costoso, sia in termini di tempo di sviluppo, sia in termini di risorse impiegate.

XAMARIN

Utilizzare Xamarin significa scrivere il codice in C#, ma compilare in codice nativo, ad esempio in Swift. Lo svantaggio è che avremo una struttura di progetto più complessa, un ambiente di sviluppo più complesso, specialmente quando si ha bisogno di compilare per iOS da un ambiente Windows e le prestazioni, sebbene buone, non sono esattamente le stesse di quelle ottenibili utilizzando il codice nativo puro. Le dimensioni dei file binari sono decenti, ma sono di solito più grandi dei file binari in codice nativo puro. Inoltre, anche

se non è tecnicamente uno svantaggio, è un errore supporre che non si dovranno imparare le complessità della piattaforma di destinazione, solo perché si sta utilizzando C#. La realtà è che bisogna anche comprendere i dettagli specifici della piattaforma, come la storicizzazione dei *keychain*, gli identificatori di *bundle app* e molto altro, ossia tutto quello che caratterizza una piattaforma di sviluppo.

XAMARIN FORMS

I *forms* in Xamarin presentano alcuni svantaggi dei quali il principale è probabilmente imputabile a *XAML*. Grazie a *XAML* è possibile costruire una UI (*user interface*) velocemente. Tuttavia un'applicazione complessa non è facile da costruire utilizzando XAML. Sicuramente i più esperti utilizzatori di XAML avranno da ridire, ma quando si tratta di produttività pura, confrontando HTML5 / CSS3 con XAML, HTML vince per distacco. Sicuramente XAML dà un controllo più sottile, ma l'HTML si adatta meglio al *"reflow"* e al multi *form*. Il reale vantaggio è quello di avere la massima portatilità con sforzo minimo e costi bassi. L'altro svantaggio dei *form* di Xamarin risiede nel fatto che il compilatore crea file molto più grandi, il che a sua volta può influire sulle prestazioni e sui tempi di caricamento. Infine, i *form* in Xamarin sono stati implementati di recente, quindi alcune funzionalità avanzate ancora non sono presenti: è una carenza che sarà comunque via via risolta con le release successive di Xamarin. Tutto ciò non vuol dire che i *form* di Xamarin non si possano utilizzare: se si desidera che un'interfaccia utente possa essere costruita con la semplicità del drag&drop, ci si accontenta di un posizionamento standard dei controlli e di una certa limitazione nel design, Xamarin fa senz'altro al caso nostro! Ad esempio, negli scenari aziendali, è frequente la necessità di sviluppare numerose applicazioni in cui è molto più importante la sostanza che l'estetica. Se invece stiamo progettando una applicazione mobile che ha bisogno di stupire, l'utilizzo di *form* di Xamarin potrebbe non essere la scelta più adatta.

CORDOVA E LE TECNOLOGIE WEB

Il più grande vantaggio di utilizzare Cordova è l'ampio accesso alle risorse, agli esempi e ai *tutorial*. Inoltre, le interfacce utente ottimizzate, create utilizzando tecnologie basate su HTML, possono essere eseguite ovunque e possono essere costruite da persone che non sono esperte in XCode, Storyboards o XAML. Un approccio progettuale in Cordova può essere: dividere il problema in piccoli blocchi, gestibili dal team, e creare il flusso di lavoro sulle esigenze e sulle competenze del team. Inoltre, con Angular e Typescript, è possibile, in modo pratico, scrivere codice mobile e farlo funzionare sul desktop e viceversa. Lo svantaggio più grande sono le performance, che sono importanti, ma non sono tutto. Analizziamo il problema da un'altra angolazione: anche le performance del tuo team sono importanti. Ad esempio, parliamo del più grande vantaggio di HTML: un'interfaccia utente cross-platform e reattiva che è attraente e facile da costruire. È vero che premendo un *div HTML* che è mascherato come un pulsante può richiedere cinque millisecondi più di, ad esempio, un pulsante nativo puro. Ma l'importante qui è che l'utente non si preoccuperebbe o addirittura non potrebbe notare cinque millisecondi di ritardo. Naturalmente, se hai fatto un pulsante da premere 100.000 volte in un ciclo, quei cinque millisecondi si aggiungono a qualcosa di sostanziale e a quel punto l'utente noterà il ritardo nella risposta dell'applicazione. I motori JavaScript funzionano abbastanza bene in una logica regolare, ad esempio sulla gestione di un'action che consente la visualizzazione di una casella di testo quando la casella di controllo è selezionata, ecc., questo tipo di azioni vengono velocemente eseguite alla stregua del codice nativo. Dove si paga un prezzo più alto è quando si inizia a mischiare tra JavaScript e codice nativo e viceversa. Questo aspetto si deve tenere presente, fin dal principio, nello sviluppo di una applicazione in Cordova, poiché l'utilizzo misto tra JavaScript e codice nativo, può portare a ritardi di alcuni secondi, che ovviamente sono misurabili dall'utente finale. L'altra situazione che si deve considerare sono gli elementi specifici dell'interfaccia utente, quali la visualizzazione di liste di scorrimento con grandi quantità di dati o applicazioni di mappe con elementi di pan e zoom con immagini molto grandi: qui è bene valutare con attenzione il bilancio fra semplificazione e prestazioni.

IL MEGLIO DEI DUE MONDI

Un punto focale qui è che ogni tecnologia ha i suoi punti di forza e debolezze e lo sviluppo delle applicazioni mobili sta cambiando rapidamente, quindi ciò che è vero oggi, nel momento in cui scrivo questo articolo, può cambiare radicalmente nel giro di un anno. La creazione dell'interfaccia UX in HTML è un approccio molto valido, fintanto che si sappia quando utilizzare HTML e JavaScript o preferire utilizzare il codice nativo. Probabilmente avrete sentito che l'utilizzo del codice nativo è costoso in termini di tempo di sviluppo e dei costi ad esso associato, e che si deve scrivere tre volte in tre diverse piattaforme. Questo è dove Xamarin entra in azione: scrivi un codice una volta, come combinazione di C # / Xamarin e HTML, CSS e JavaScript. Con il giusto mix si possono avere delle prestazioni simili a quelle del codice nativo. Un risultato davvero notevole!

XAMARIN

Con un codice condiviso basato su C#, gli sviluppatori possono usare gli strumenti Xamarin per scrivere applicazioni native Android, iOS e Windows con interfacce utenti native e condividere il codice su diverse piattaforme. Xamarin ha oltre 1 milione di sviluppatori distribuito in più di 120 paesi nel mondo (Maggio 2015). Nel febbraio del 2016, Microsoft ha acquistato l'azienda e, dopo qualche mese, ha reso i prodotti *Xamarin Studio* gratuiti per sviluppatori e studenti.

NOTA

APACHE CORDOVA

Prima noto come **PhoneGap**, è un framework per lo sviluppo di applicativi mobili. Apache Cordova permette ai programmatori di creare applicazioni mobili usando CSS3, HTML5 e JavaScript invece di affidarsi ad API specifiche delle piattaforme Android, iOS o Windows Phone. Il framework incapsula poi il codice CSS, HTML e JavaScript generato all'interno delle predette piattaforme. Le applicazioni generate dal framework non possono né considerarsi puramente native (il rendering della struttura grafica è fatto con visualizzazioni Web) né basate completamente sul Web (Il programma viene impacchettato come una applicazione per la distribuzione e hanno accesso alle API native dei dispositivi mobili). Mescolare snippet di codice nativo e ibrido è possibile dalla versione 1.9.

LA NOSTRA APP

Cerchiamo di approcciare alcuni aspetti di Cordova e Xamarin attraverso un esempio pratico. Proviamo a creare un'applicazione che inserisce i dati in un'API Web scritta in ASP.NET, focalizziamoci solo sugli aspetti principali dei framework, manteniamo il WebAPI estremamente semplice. Non esiste alcuna autenticazione e non esiste un carico di lavoro complesso. È solo un'azione semplice che accetta una stringa in POST HTTP. Ciò riflette uno scenario reale, in cui l'applicazione ha un grande carico di dati da pubblicare. L'applicazione non pubblicherà l'intero set di dati in una sola richiesta. Questo approccio non solo ha implicazioni positive sulla memoria del dispositivo mobile, ma evita anche il time-out dell'applicazione. Per raggiungere lo scopo, separiamo la richiesta in blocchi più piccoli. L'api Web può essere visualizzata nel listato 1.

Codice 1:

```
public class TestController : ApiController
{
  public void SubmitData(string inputData)
  {
    System.Diagnostics.Debug.WriteLine(inputData);
  }
}
```

Questa API può essere chiamata in un URL come questo:

```
/api/test/SubmitData?inputData="sampleinput"
```

Definita la chiamata API, scriviamo l'app. L'applicazione svolge un compito molto semplice: chiama l'API 10.000 volte. Misureremo il tempo necessario per inviare tali richieste. Da notare che non siamo interessati a misurare quanto tempo ha effettivamente preso il server per rispondere. Interessa solo il funzionamento di queste richieste. Il motivo per cui si desidera misurare il tempo impiegato solo per inviare le richieste è un modo per poter ottenere un confronto realistico tra i linguaggi nativi e JavaScript, e vedere se c'è effettivamente necessità di utilizzare i linguaggi nativi, dando così ragione ai programmatori puristi. Non andremo a misurare il traffico di rete o altri fattori estranei che influenzino questa misurazione.

CORDOVA APP

Una minima conoscenza di Cordova è richiesta, da qui in poi: creiamo un progetto base Cordova e aggiungiamo *jQuery*. Abbiamo ripulito l'interfaccia utente per rimuovere quel messaggio pronto per il dispositivo che mostra l'applicazione Starter Cordova e aggiungiamo invece un pulsante e un div, come mostrato in questo snippet:

Fig. 1: L'app Cordova: il tempo di invio delle richieste

```
<div id="deviceready">
  <br/>
  <button id="sendDataButton">
   Send Lots of Data</button>
  <div id="sendDataButtonLog"></div>
</div>
```

Inoltre, poiché vogliamo effettuare chiamate HTTP dall'applicazione, bisogna aggiungere quanto segue al meta tag *"Content-Security-Policy"*:

```
;connect-src http://*/*
```

Si noti che quanto sopra è molto insicuro: consente di effettuare chiamate a qualsiasi URL HTTP. Va bene per scopi di prova. Successivamente, modifichiamo la parte JavaScript del codice. L'idea è che quando si fa clic su *sendDataButton*, si desidera eseguire 10.000 richieste POST e misurare quanto tempo è stato necessario per inviare tali richieste. Non siamo interessati a quanto tempo impiegano queste richieste a tornare. La prima cosa da fare qui è aggiungere un gestore eventi per l'evento che viene invocato da *sendDataButton*. È possibile eseguire questa operazione nella funzione di inizializzazione, come mostrato in questo ulteriore

frammento:

```
$("#sendDataButton").bind('click', app.sendData);
```

Come potete vedere, facendo clic sul pulsante chiama una funzione *sendData* nell'oggetto app. Questa funzione può essere visualizzata nel Codice 2:

```
sendData:function() {
    var now = window.performance.now();
    var deferreds = [];
    var i = 1;
    for (i = 1; i <= 10000; i++) {
        deferreds.push(sendDataChunk());
    }
    var then = window.performance.now();
    document.getElementById(
     "sendDataButtonLog").innerHTML =
     "Total time (ms): " + (then - now);

    $.when.apply($, deferreds).done(function() {
        // you can check for total time
        // for request to complete here.
    });
    function sendDataChunk() {
        var deferred = new $.Deferred();
            var postUrl = "..removed..";
            $.ajax({
                type: "POST",
                url: postUrl,
                success: function() {
                    deferred.resolve();
                }
            });
        return deferred;
    }
}
```

Come mostra il Codice 2, stiamo lanciando 10.000 richieste, ma non aspettiamo di ricevere la risposta. Siamo semplicemente interessati a misurare il tempo necessario per inviare le richieste. In questo modo, si passa da *JavaScript* a codice nativo, anche se tutto questo viene implementato nel *WebBrowser* stesso. Potete immaginare che utilizzare la *shell* di Cordova sarebbe probabilmente un'operazione ancora più costosa. Un esempio di utilizzo della *shell* di Cordova potrebbe essere: chiamare l'API della fotocamera, o forse un'account di archiviazione dati offline, escluso *window.localStorage*. Eseguiamo l'applicazione: viene mostrata un'interfaccia utente con un pulsante che dice *"Send Lot of Data"*. Premiamo quel pulsante e l'applicazione ci informa quanto tempo ci è voluto per inviare 10.000 richieste. 10.000 richieste sono abbastanza, quindi,

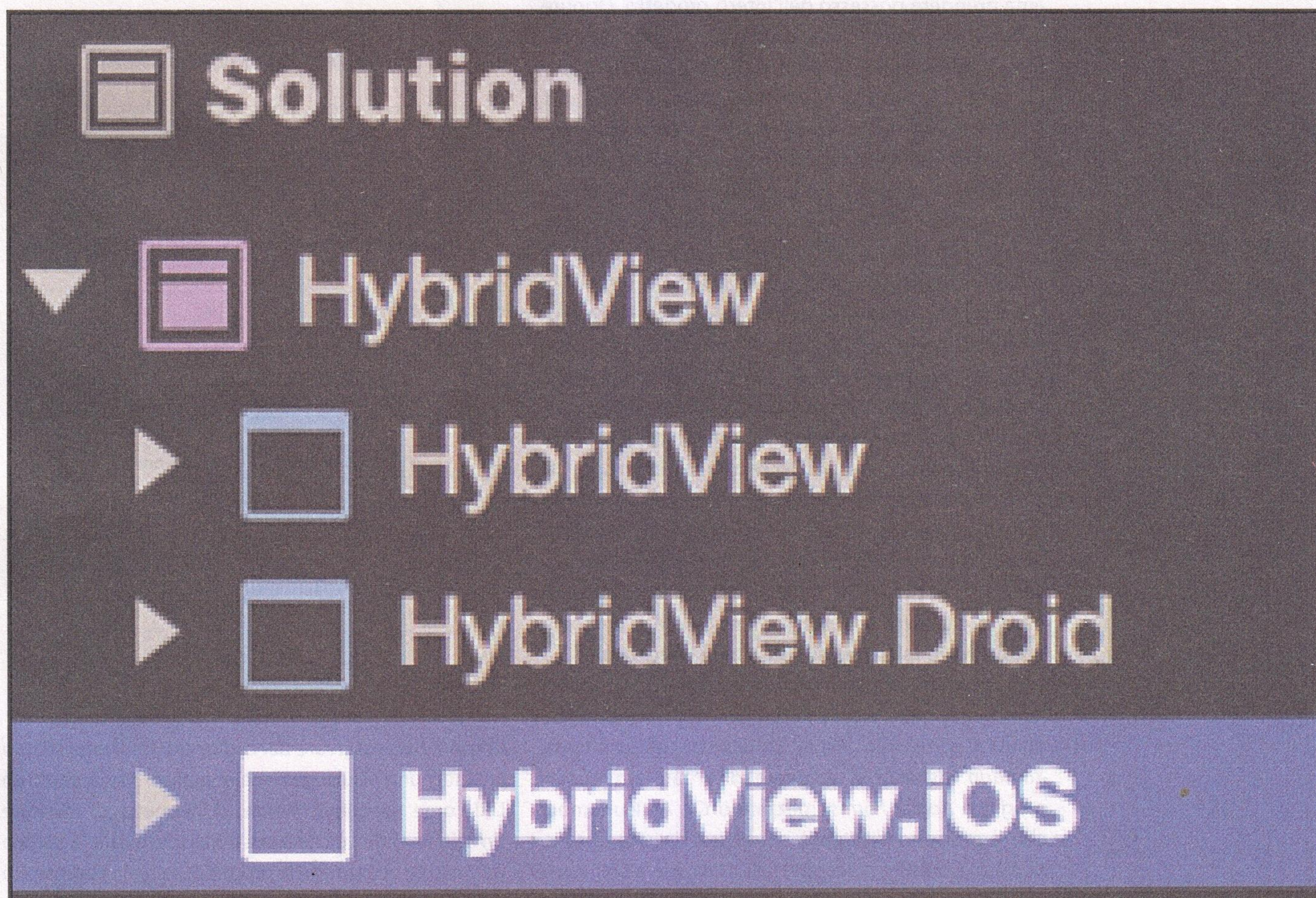



Fig. 2: La struttura del progetto della App in Xamarin

come si può vedere dalla Figura 1, ci sono voluti quasi 14 secondi per inviarle. Ora ripetiamo questo esempio utilizzando il codice nativo scritto in **Xamarin**.

UNA APP XAMARIN "IBRIDA"

Invece di scrivere un'applicazione completa basata su moduli *Xamarin*, preferisco invece scrivere un'applicazione ibrida. Un'applicazione ibrida, in questo caso, è un'applicazione *Xamarin* che utilizza tecnologie basate sul Web per l'interfaccia utente. In questo modo si ha la possibilità di utilizzare il codice nativo di *Xamarin* per eseguire un carico di lavoro pesante ovunque sia necessario. Ad esempio, qui chiediamo al codice nativo di fare le richieste POST per noi. Questo è un approccio molto funzionale, perché possiamo andare avanti e indietro tra le applicazioni di Cordova e le applicazioni personalizzate *Xamarin Hybrid*.

Come comportarsi con i plug-in? I plug-in sono un modo per raggiungere l'obiettivo, ma utilizzare un plug-in, di solito, significa ancora che bisogna riscrivere il plug-in in più linguaggi nativi, per ogni singola piattaforma. Questo ci riporta al problema originale: evitare più codici sorgente. Potremmo scrivere un plug-in utilizzando *Xamarin* ma, *Xamarin* ama prendere possesso dell'intero progetto. Oppure potremmo contare sulla comunità ed utilizzare dei plug-in già sviluppati. Ma questi plug-in sono scritti per ambiti generici. Per esempio, immaginiamo che il requisito sia quello di cercare all'interno un negozio offline, utilizzando una chiave di ricerca. Potrei utilizzare un plug-in per leggere l'intero contenuto del negozio offline, quindi eseguirlo in un ciclo, passare attraverso il firewall di JavaScript nativo più e più volte e pagare pesantemente in termini di prestazioni. Comunque questo è un esempio costruito a tavolino e per tali scenari comuni esistono dei plug-in che consentono di passare una sola volta la chiave di ricerca. Diciamo che poiché l'iterazione è eseguita puramente in codice nativo, è abbastanza verosimile che non esista un plug-in che corrisponda al requisito specifico di business. Potremmo scriverlo con *ObjectiveC*, *Java*, *WinJS*, e etc! Oppure, possiamo evitare questa complessità e utilizzare *Xamarin* per i linguaggi nativi, *HTML* e *JavaScript* per UI e logica di base di business. Questo sembra essere il percorso più breve, in relazione allo sforzo da effettuare.

Iniziamo: la prima cosa da fare è creare un progetto *Xamarin*. Avviare Visual Studio per Mac (se si dispone di un Mac) o solo Visual Studio con gli strumenti *Xamarin* installati e scegliere di creare un nuovo progetto *Applet Forms* utilizzando *C#*. Assicuriamoci che il progetto di codice condiviso sia una libreria di classi portatile e scegliere di utilizzare *XAML* per i file di interfaccia utente. Chiamiamo questo progetto *HybridView*. Una volta creato il progetto, si dovrebbe vedere una struttura del progetto come quella mostrata in Figura 2. Successivamente, nel progetto *HybridView*, aggiungere una nuova classe denominata *HybridWebView* e inserire il codice riportato nel Codice 3.

```
using System;
using Xamarin.Forms;

namespace HybridView
{
    public class HybridWebVi ew : View
    {
        Action<string> action;
        public static readonly BindableProperty
                                        UriProperty =
        BindableProperty.Create(
            propertyName: "Uri",
            returnType: typeof(string),
            declaringType: typeof(HybridWebView),
            defaultValue: default(string));

        public string Uri
        {
            get { return (string)GetValue(UriProperty);}
              set { SetValue(UriProperty, value) ;}
        }

        public void RegisterAction(Action<string>
                                            callback)
        {
            action = callback;
        }

        public void Cleanup()
        {
            action = null;
        }

        public void InvokeAction(string data)
        {
            if (action == null || data == null)
            {
                return;
            }
            action.Invoke(data);
        }
    }
}
```

Come si può osservare dal codice qui riportato, stiamo creando una visualizzazione semplice che accetta un URI come proprietà. Inoltre, ci permette di invocare un'azione. L'idea è che è possibile specificare la pagina da caricare utilizzando la proprietà e che la pagina può richiamare la funzionalità del codice nativo utilizzando l'azione. Successivamente, utilizzare questa vista nel

ContentPage di HybridViewPage. Andiamo avanti e modifichiamo la sua porzione XAML, come mostrato qui di seguito (Codice 4):

```
<?xml version="1.0" encoding="utf-8"?>
<ContentPage
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/
                                winfx/2009/xaml"
    xmlns:local="clr-namespace:HybridView"
    x:Class="HybridView.HybridViewPage">
  <ContentPage.Content>
    <local:HybridWebView x:Name="hybridWebView"
                                Uri="index.html"
        HorizontalOptions="FillAndExpand"
        VerticalOptions="FillAndExpand" />
  </ContentPage.Content>
</ContentPage>
```

Come si può vedere, stiamo utilizzando la vista creata e la indichiamo in un file chiamato *index.html*. Creiamo questo file nei progetti *HybridView.iOS/Droid* e lo inseriamo come contenuto. Successivamente, è necessario gestire l'azione che può essere invocata dal codice JavaScript e nel codice nativo, eseguire 10.000 richieste *POST* e misurare il tempo necessario per inviare tali richieste. Come prima, non aspettiamo che tornino le risposte. Questo codice va nel file *HybridViewPage.xaml.cs* e può essere visualizzato nel Codice5:

```
using System.Net;
using Xamarin.Forms;

namespace HybridView
{
  public partial class HybridViewPage : ContentPage
  {
    public HybridViewPage()
    {
      InitializeComponent();
      hybridWebView.RegisterAction(data =>
                                    sendData());
    }

    private void sendData()
    {
      var now = System.DateTime.Now;
      string url = "..removed..";
      for (int i = 0; i < 10000; i++)
      {
        HttpWebRequest request =
            (HttpWebRequest)WebRequest.
                                    Create(url);
        request.Method = "POST";
        request.BeginGetResponse(
            (ar) =>
```

```
          System.Diagnostics.Debug.WriteLine(
            "end:" + ar.AsyncState.ToString()),
                                            i);
      }
      var then = System.DateTime.Now;
      DisplayAlert("Alert",
          "Total time (ms):" +
          (then - now).TotalMilliseconds,
          "OK");
    }
  }
}
```

Perfetto! Tutto quello che resta da fare è creare un semplice file *index.html* che può chiamare questa azione e un rendering che può caricare questa *HybridWebView*. Per prima cosa creiamo il rendering di visualizzazione come una classe nel progetto *HybridView.iOS*:

```
using System.IO;
using HybridView;
using HybridView.iOS;
using Foundation;
using WebKit;
using Xamarin.Forms;
using Xamarin.Forms.Platform.iOS;

[assembly: ExportRenderer(typeof(HybridWebView),
typeof(HybridWebViewRenderer))]
namespace HybridView.iOS
{
  public class HybridWebViewRenderer :
  ViewRenderer<HybridWebView, WKWebView>,
                          IWKScriptMessageHandler
  {
    const string JavaScriptFunction =
    "function invokeNativeCode(data){
    window.webkit.messageHandlers.invokeAction.
                          postMessage(data);}";
    WKUserContentController userController;

    protected override void OnElementChanged(
      ElementChangedEventArgs<HybridWebView> e)
    {
      base.OnElementChanged(e);

      if (Control == null)
      {
        userController = new WKUserContentController();
        var script = new WKUserScript(
        new NSString(JavaScriptFunction),
         WKUserScriptInjectionTime.AtDocumentEnd,
                                          false);
         userController.AddUserScript(script);
        userController.AddScriptMessageHandler(this,
        "invokeAction");

        var config = new WKWebViewConfiguration {
```

JAVASCRIPT

È un linguaggio di scripting orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione Web lato client per la creazione, in siti Web e applicazioni Web, di effetti dinamici interattivi tramite funzioni di script invocate da *eventi* innescati a loro volta in vari modi dall'utente sulla pagina Internet in uso (mouse, tastiera, caricamento della pagina ecc...).

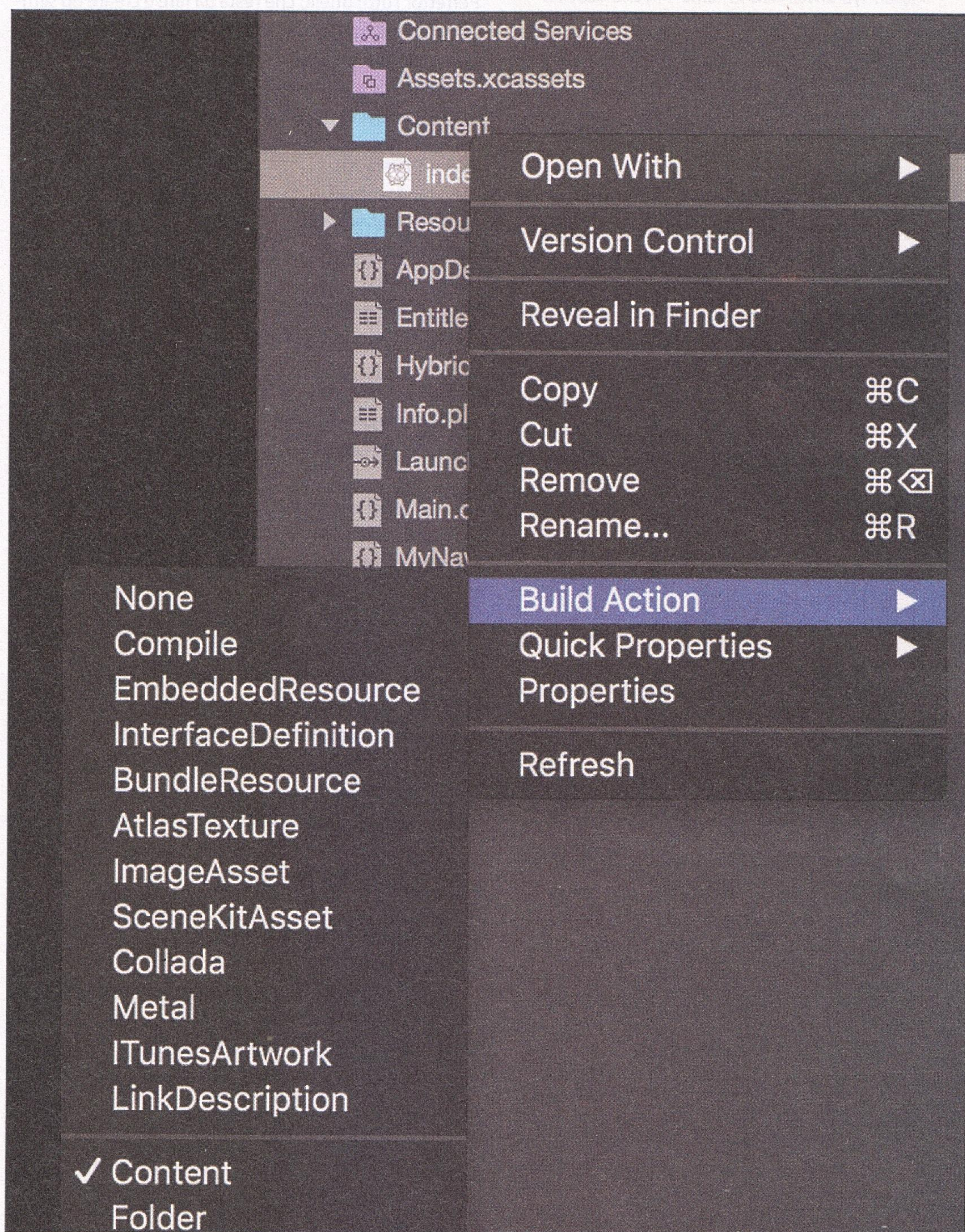
```
        UserContentController = userController };
        var webView = new WKWebView(Frame, config);
        SetNativeControl(webView);
    }
    if (e.OldElement != null)
    {
        userController.RemoveAllUserScripts();
        userController.RemoveScriptMessageHandler
                                ("invokeAction");
        var hybridWebView = e.OldElement as
                                HybridWebView;
        hybridWebView.Cleanup();
    }
    if (e.NewElement != null)
    {
        string fileName =
            Path.Combine(NSBundle.MainBundle.BundlePath,
            string.Format("Content/{0}", Element.Uri));
            Control.LoadRequest(
                new NSUrlRequest(new NSUrl(fileName,
                                            false)));
    }
}

public void DidReceiveScriptMessage(
    WKUserContentController userContentController,
```


Fig. 3: Costruzione dell'action per il file index.html

```
WKScriptMessage message)
    {
        Element.InvokeAction(message.Body.ToString());
    }
  }
}
```

Per l'*index.html*, creiamo una cartella denominata *Content* nel progetto *HybridView.iOS* e copiamo il file *index.html* in quella cartella. Assicuriamoci che l'azione di creazione sia impostata su *Content*, come si può vedere nella figura 3.

Non resta che scrivere del semplice codice nel file HTML che chiama la funzione nativa:

```
<button type="button"
  onclick="invokeNativeCode('hello');">
  Invoke Native Code
</button>
```

Andiamo avanti e creiamo questa app, facendola girare nello stesso emulatore o nel dispositivo in cui abbiamo eseguito il progetto Cordova. L'interfaccia utente mostra un pulsante che è possibile cliccare. Andiamo avanti e clicchiamoci su! Così facendo si inviano 10.000 richieste POST e viene mostrato il tempo necessario per inviare tali richieste, come si può vedere in Figura 4.
Come è chiaro, l'applicazione Xamarin, essendo nativa, è più veloce dell'app Cordova. Le esecuzioni successive mostrano che l'applicazione Xamarin per questo compito specifico è quattro o 15 volte più veloce. Questo tipo di differenza è quella che l'utente finale può notare nell'eseguire una applicazione. Consideriamo però che stiamo ancora utilizzando un'interfaccia utente basata su HTML, per questa ragione si paga un prezzo in termini di prestazioni, ma l'utente non nota il ritardo di cinque millisecondi necessari per fare clic su un pulsante HTML rispetto a un pulsante nativo. Infatti, in molti casi, un UI HTML può funzionare meglio di un UI nativo. Ma il punto principale qui è che il fatto di essere in grado di scegliere e combinare diverse piattaforme e divere tecnologie, ci dà una grande flessibilità nel costruire una interfaccia utente responsive ed efficiente. Quando è necessario "sporcarsi le mani" col codice nativo, possiamo farlo utilizzando Xamarin. In questo modo abbiamo il meglio di tutte le soluzioni.

CONCLUSIONI

In una terra lontana, lontana, c'è una battaglia tra gli sviluppatori. Alcuni insistono sul fatto che il codice nativo è l'unico modo per scrivere applicazioni. Altri insistono sul fatto che la creazione di app utilizzando codice nativo offre scarsi vantaggi e la creazione di applicazioni Cordova è il modo giusto per soddisfare le esigenze dell'utente. Per fortuna vivo lontano da quella terra di battaglie. Ritengo che il paesaggio dello

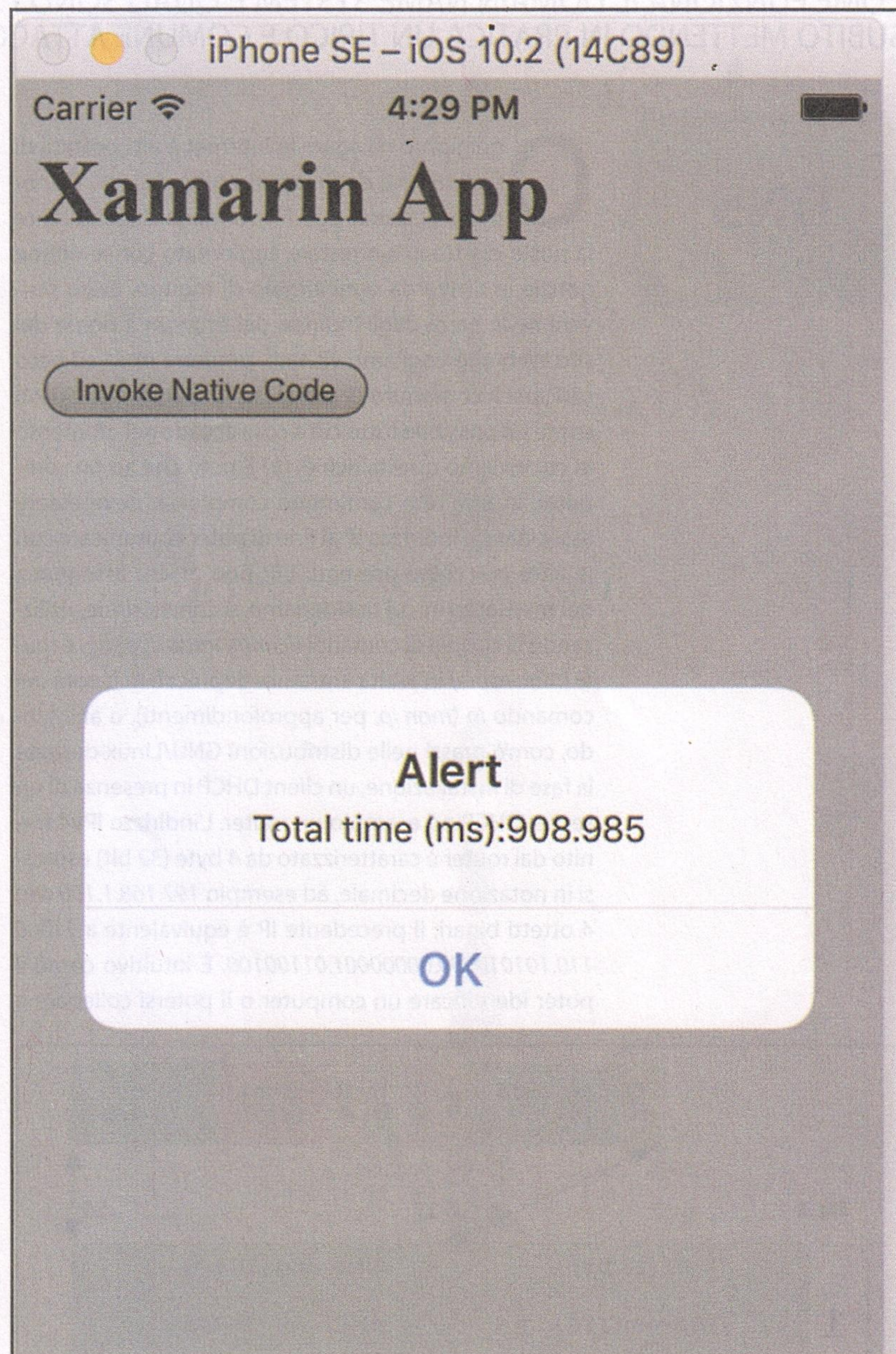



Fig. 4: L'app Xamarin esegue lo stesso lavoro dell'app Cordova

sviluppo legato al mobile cambia molto rapidamente e invita a tenersi sempre aggiornato sulle novità in modo da sapere sempre qual è il modo migliore per creare un'applicazione oggi. Dal mio punto di vista, in questo momento, scrivere le interfacce utente in HTML è il modo più semplice. Il codice nativo, in linea generale offre delle prestazioni migliori, ma i casi in cui tali prestazioni sono apprezzabili sono rari. Per fortuna, è possibile scegliere lo strumento giusto in base al progetto. A volte è Xamarin, a volte è Cordova, e a volte il codice nativo.

DNS: È DAVVERO SICURO?

COME FUNZIONA IL DOMAIN NAME SYSTEM E QUALI SONO I SUOI PUNTI DEBOLI? SCOPRIAMOLO SUBITO METTENDO IN PRATICA UN TIPICO E COMUNE ATTACCO

Oggigiorno navigare in Internet è alla portata di tutti, anche degli utenti meno esperti. Dopotutto, nessuno può fare a meno di controllare la posta elettronica o restare aggiornato con le ultime notizie in arrivo da ogni angolo di mondo. Basta scrivere nella barra degli indirizzi del browser il nome del sito Web che vogliamo visitare, premere *Invio* ed ecco caricarsi il contenuto di interesse. Ci siamo mai chiesti come sia possibile tutto ciò e cosa accade nel momento in cui inviamo questa richiesta? È noto che ad un computer, in una rete comunque complessa, deve essere associato un indirizzo IP al fine di poter comunicare con le altre macchine presenti. L'IP può essere assegnato, nel momento in cui desideriamo la connessione, utilizzando la coppia di comandi *ifconfig (man ifconfig)* e *route (man route)* in realtà entrambi deprecati in favore del comando *ip* (*man ip*, per approfondimenti), o abilitando, com'è prassi nelle distribuzioni GNU/Linux durante la fase di installazione, un client DHCP in presenza di un server DHCP, ad esempio un router. L'indirizzo IPv4 fornito dal router è caratterizzato da 4 byte (32 bit) espressi in notazione decimale, ad esempio *192.168.1.100* o in 4 ottetti binari: il precedente IP è equivalente a *11000 110.10101 0000.00000001.01100100*. È intuitivo come il poter identificare un computer o il potersi collegare a quel determinato server utilizzando un nome facile da ricordare, renda le operazioni sicuramente molto più semplici dal punto di vista dell'utente.

IL PROTOCOLLO

Acronimo di Domain Name System, il DNS è il servizio di traduzioni dei nomi che associa un nome simbolico al corrispondente indirizzo IP. Si compone di 3 parti: un protocollo al livello Applicazione della pila ISO/OSI per lo scambio messaggi tra client e server, una gerarchia di server dei nomi e un database di nomi in esso distribuiti. Perché non pensare ad un sistema centralizzato per fornire il servizio DNS? Mancanza di ridondanza in caso di guasti, volumi di traffico (numero di richieste) che graverebbero solo su una macchina, database centralizzato con richieste più lente nei punti distali e in caso di manutenzione andrebbe giù il servizio. In poche parole, assenza di scalabilità. Per questo motivo si è scelto un servizio distribuito. Il primo elemento che analizzeremo è il protocollo.

In **fig. 1** è visibile il formato del messaggio, le informazioni che si scambiano client e server che permettono agli host di interrogare il database distribuito per risolvere i nomi. Consta di una parte fissa (header o intestazione) di dimensione 12 byte suddivisa in 6 campi da 2 byte ed una parte di lunghezza variabile contenente informazioni e messaggi di risposta alla richiesta del client. Iniziamo dalla parte variabile non prima di aver ricordato che in informatica un *Resource Record (RR)* indica un insieme di dati strutturati in campi: ad esempio, in un database un record corrisponde alla riga di una tabella e nello specifico ad una voce del database distribuito. Premesso ciò, la parte variabile presenta 4 sezioni. La prima, *Question Section*, contiene informazioni sull'interrogazione (*DNS query*) inoltrata dal client: ad esempio, nome e tipo. L'indirizzo IP di un hostname equivale ad una richiesta di tipo A (abbreviativo di *a host address*), mentre si ha una richiesta di tipo *MX* (*Mail eXchange*) qualora si richieda un'interrogazione a un server di posta elettronica di un dato dominio, come da specifiche del protocollo *SMTP* (*Simple Mail Transfer Protocol*). Segue la sezione *Answer Section* presente nel messaggio di risposta: poiché ad un hostname possono corrispondere più indirizzi IP, viene specificato il numero di RR attinenti. Contenuta nel messaggio di risposta anche la sezione *Authority Section*: specifica il numero di RR relativi ad altri server (ad esempio, chi sia il ser-

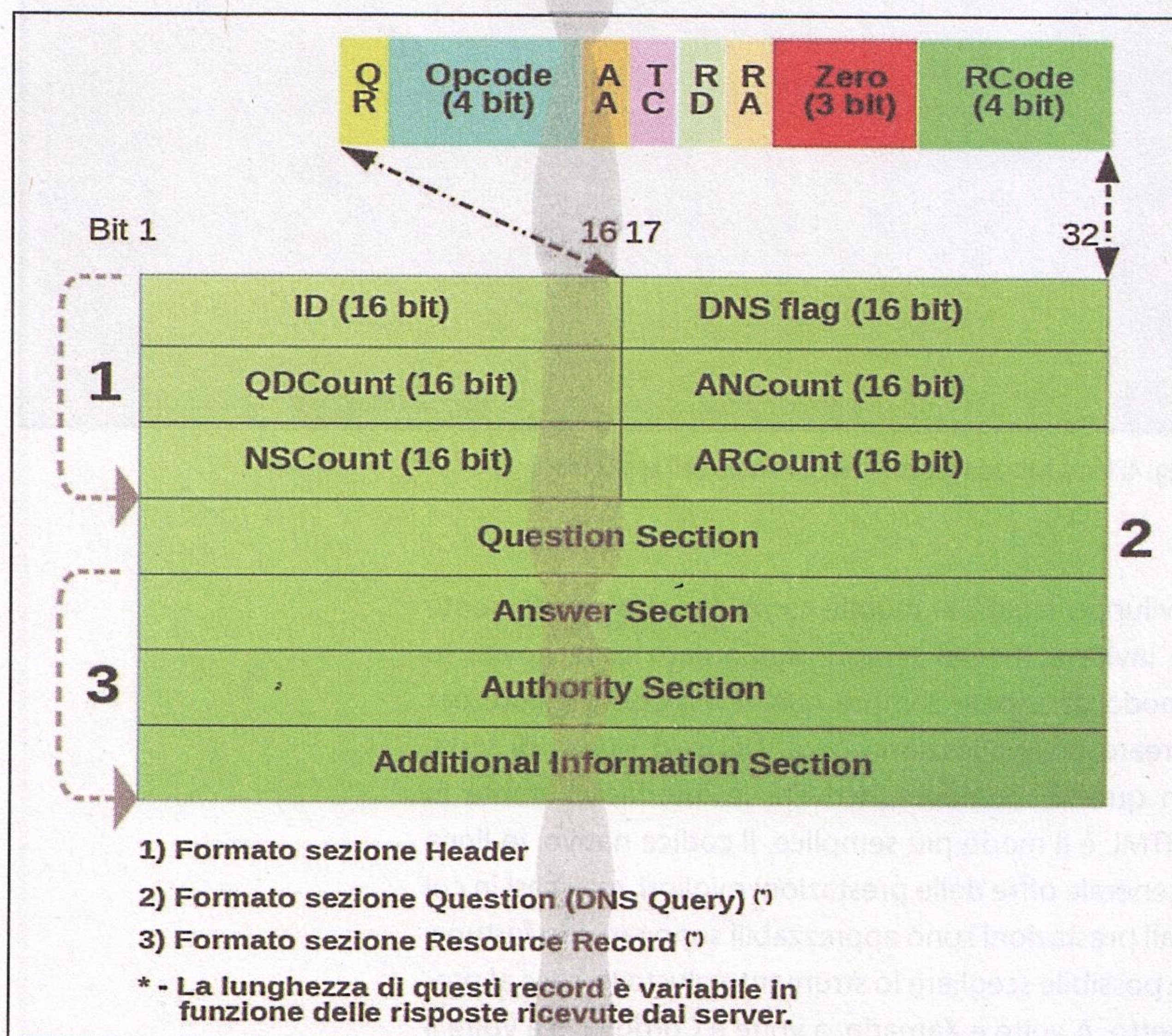



Fig.2: Il formato del messaggio nel protocollo DNS

ver autoritativo per quel particolare dominio). Anche l'ultima sezione, *Additional Information Section*, è presente nel messaggio di risposta: specifica ulteriori RR, ad esempio in risposta ad un'interrogazione di tipo MX conterrà un record di tipo A che riporterà l'indirizzo IP del server di posta. Il client, attraverso i campi della parte fissa, riconosce il tipo di richiesta e di cosa si compone. Infatti, il campo identification (*ID*) è un numero che identifica la richiesta e viene copiato tal quale anche nel messaggio di risposta dal server affinché il client possa associare quella risposta alla specifica richiesta fatta in precedenza. Segue 1 bit per la *QR* (*Question Response*): se 0 è una richiesta del client, il valore 1 una risposta del server. I successivi 4 bit definiscono un *Opcode* specificando il tipo di query che il messaggio sta portando: il campo è impostato dal creatore della query e copiato immutato nella risposta. Per le flag che seguono verranno specificati alcuni termini che al momento ancora non conosciamo ma che analizzeremo a breve. Il flag da 1 bit *AA* (*Authoritative Answer*) se di valore 1 indica che il server che ha dato la risposta è autoritativo per quel dominio, altrimenti è una risposta non autoritativa. Il bit successivo è riservato al flag *TC* (*Truncation Flag*): l'invio del messaggio avviene via UDP la cui lunghezza limite è 512 byte e oltre la quale avviene un troncamento. Se il flag TC è 1 vuol dire che il messaggio è stato troncato: in questi casi il client potrebbe necessitare di stabilire una connessione TCP in luogo di UDP poiché TCP non ha un limite sulla lunghezza dei messaggi.

Il bit successivo è il flag *RD* (*Recursion Desired*) a cui fa seguito 1 bit del flag *RA* (*Recursion Available*). La prima impostata a 1 richiede al server una risoluzione ricorsiva all'interrogazione. La seconda, impostata dal server, se pari a 1 dice al client che è supportata la risoluzione ricorsiva. I primi 4 byte della parte fissa terminano con i rimanenti 7 bit di cui i primi 3 riservati e i rimanenti 4, *RCode* (*Response Code*), forniscono un codice che in decimale è compreso tra 0 (0000 in binario) e 10 (1010 in binario).

Questi 4 bit vengono impostati a zero dal client all'atto della richiesta e cambiati dal server in funzione del risultato della richiesta. Così, ad esempio, se rimangono invariati al valore 0, vuol dire che nessun errore è avvenuto. Qualsiasi altro valore fornisce un'indicazione ben precisa: ad esempio, 4 (0100 in binario) indica che il tipo di interrogazione ricevuta non è supportata dal server o 9 (1001 in binario) che il server non è autoritativo per la zona specificata. Per approfondimenti si suggerisce di iniziare la lettura dell'RFC 1035 (https://tools.ietf.org/html/rfc1035). La parte fissa si compone ancora di 4 campi da 2 byte (16 bit) ognuno dei quali fornisce indicazioni sulle occorrenze presenti nella sezione variabile. Il primo campo è *QDCount* (*Question Count*) che riporta il numero di richieste presenti nella sezione Question Section. Segue *ANCount* (*Answer Count*): numero di RR presenti nella sezione Answer Section. Il campo *NSCount* (*Name Server Count*) specifica quanti RR sono presenti nella Authority Section e infine *ARCount* (*Additional Record Count*) specifica il numero di RR riportati nella sezione *Addictional Information Section*. Facciamo presente che la porta utilizzata dal protocollo è la 53, in genere su *UDP* (*User Datagram Protocol*) a causa del minore overhead (sovraccarico): i messaggi sono corti e un solo messaggio deve essere trasferito dal client al server e viceversa.

ECCO COME FUNZIONA

Il protocollo DNS è utilizzato dagli altri protocolli del livello applicativo (HTTP, SMTP e FTP) come base per il proprio funzionamento: nello specifico per tradurre l'hostname in indirizzi IP. I nameserver sono i server che rispondono alle richieste di messaggi del protocollo DNS. La **fig. 2** riporta una possibile dinamica. Si ipotizza che l'azienda abbia un server DNS: non tutte lo hanno e chi ce l'ha è in genere un *DNS caching-only*, non risolve i nomi, memorizza solo le richieste in modo che richieste uguali ad opera di altri client vengano soddisfatte senza la necessità di inoltrare una nuova richiesta DNS. Se non si possiede un server DNS interno si può sempre specificarne uno esterno previo uso del file */etc/resolv.conf*. In una "connessione domestica" questo file è governato dal Network Manager, il quale riporta come DNS l'indirizzo IP del nostro router dove sono configurati gli indirizzi IP dei DNS primario e secondario del nostro provider. Il funzionamento del DNS è basato su una suddivisione gerarchica e distribuita dello spazio dei nomi (domini) organizzati in livelli con una struttura ad albero (**fig. 3**).

I server che rispondono alle richieste dei client sono organizzati – al di là di server locali aziendali – principalmente in 3 classi: Root, TLD (Top-Level Domain) e SLD (Second-Level Domain). I nomi dei domini vengono indicati da stringhe separate da punti a costituire l'indirizzo completo

Fig. 2: Seguire la numerazione per la dinamica di risoluzione del nome

NOTA

COME FUNZIONAVA AGLI ALBORI

A partire dal 1983 Paul Mockapetris e Jon Postel eseguivano i loro primi test sul DNS. Fino ad allora la risoluzione dei nomi era affidata ad un file da scaricare e aggiornare periodicamente all'interno del quale erano riportate le associazioni indirizzi IP-nomi. Stiamo parlando delle origini di Internet quando le macchine erano poche e la rete scarsamente estesa. Il file in questione era (ed è presente tutt'oggi) /etc/hosts. Con il crescere dell'estensione della rete e del numero delle macchine questo approccio non andava più bene poiché occorreva ogni volta provvedere ad un aggiornamento, anche svariate volte al giorno e non sempre risultava completo. Per questo motivo venne creato il protocollo DNS con la struttura gerarchica riportata in queste pagine. Il file hosts oggi svolge una funzione simile ad una "rubrica".

NOTA

CACHE, SERVER E RESOLVER

Quali sono le differenze?

Il termine DNS cache può indicare due situazioni distinte fra loro. Una lista di nomi che il server ha risolto di recente e che mantiene in memoria (di default per un tempo minimo di 3 ore) o un server DNS non autoritativo che esegue richieste ricorsive che poi memorizza in cache. Un DNS Server è un software, tra i più utilizzati in ambito Open Source c'è BIND (Berkeley Internet Name Domain, maggiori informazioni disponibili all'indirizzo Web www.isc.org/downloads/bind/), che ha il compito di risolvere le richieste dei client: di fatto è la parte software installata nei server DNS. Infine, un DNS Resolver è un programma (librerie standard C), ad esempio chiamato in causa dal browser, che agisce da client per interrogare il server DNS secondo due modalità: ricorsiva o iterativa.

(*FQDN - Fully Qualified Domain Name*): la gerarchia si sviluppa partendo da destra verso sinistra. Ad esempio, per il dominio *www.edmaster.it* il primo valore è *.it* e costituisce il TLD caratterizzato da nome generici (.com, .net, ecc) e da nomi di nazioni (.it, .de, .au, ecc). Il meccanismo della delegazione attua la decentralizzazione delle informazioni. Per ogni livello deve esserci un server DNS responsabile della risoluzione dei nomi del ramo d'albero sottostante, che per questo è detto autoritativo. A causa della struttura gerarchica dei nomi potrebbe accadere che il server responsabile per un dominio di un certo livello deleghi un altro server per la risoluzione del dominio del livello successivo, server che a quel punto diventerà autoritativi per quel sotto-dominio. Così facendo si crea la gerarchia di cui sopra in maniera tale da poter proseguire senza alcun intoppo fino a che non si incontra il server DNS autoritativo che conosce la risposta finale.

Con riferimento alla **fig. 2**, nella richiesta di risoluzione di un nome da un browser, quindi da un client HTTP, il primo server ad essere interrogato è il local name server aziendale. Al posto del server DNS aziendale possiamo anche immaginare il server DNS del nostro provider: se questa macchina non può soddisfare la richiesta e non sa come contattare il server autoritativo del dominio, ecco che deve iniziare una ricerca partendo con una richiesta indirizzata al server Root. A questo punto, il server aziendale (o del provider) diventa a tutti gli effetti un client con il fine di portare a termine la richiesta del client iniziale. I server Root dislocati nel mondo sono 13 ognuno dei quali è replicato, per motivi di sicurezza e ridondanza, diverse volte per un totale di 247 server. Se il server root interrogato ha la richiesta la invia altrimenti, poiché il primo valore del dominio www.edmaster.it è .it, viene ritornato l'indirizzo IP del server TLD che è autoritativo per i nomi .it. A questo punto il server locale (o del provider) contatta il server autoritativo per i domini .it il quale non può rispondere alla richiesta (a meno di averla già in cache) e quindi invia l'indirizzo IP del server DNS che è autoritativo per i domini di secondo livello.

Solo a questo punto il server aziendale (o del provider) si vede arrivare l'indirizzo IP del nome che abbiamo immesso nella barra degli indirizzi del browser e può così soddisfare alla richiesta girandola al client iniziale. I server autoritativi (o di competenza) altro non sono che organizzazioni dotate di host Internet pubblicamente accessibili (come i Web server e/o i server di posta) le quali forniscono record DNS di pubblico dominio al fine di mappare il nome dell'host in un indirizzo IP: possono essere mantenuti dall'organizzazione (università, istituti di ricerca, ecc) o appoggiarsi ad un service provider. Facciamo presente che l'utente (client) che effettua la richiesta non ha visibilità di tutta la serie di passaggi accennati: le operazioni avvengono in maniera assolutamente trasparente e in pochi millesimi di secondo!

PROBLEMI CON IL PROTOCOLLO DNS

Ora che conosciamo, almeno come principio di funzionamento, la dinamica del servizio DNS poniamo subito una domanda: notato nulla di strano? Ebbene sì, analogamente al protocollo DHCP e ARP, anche il protocollo DNS, nonostante il suo ruolo essenziale per il funzionamento

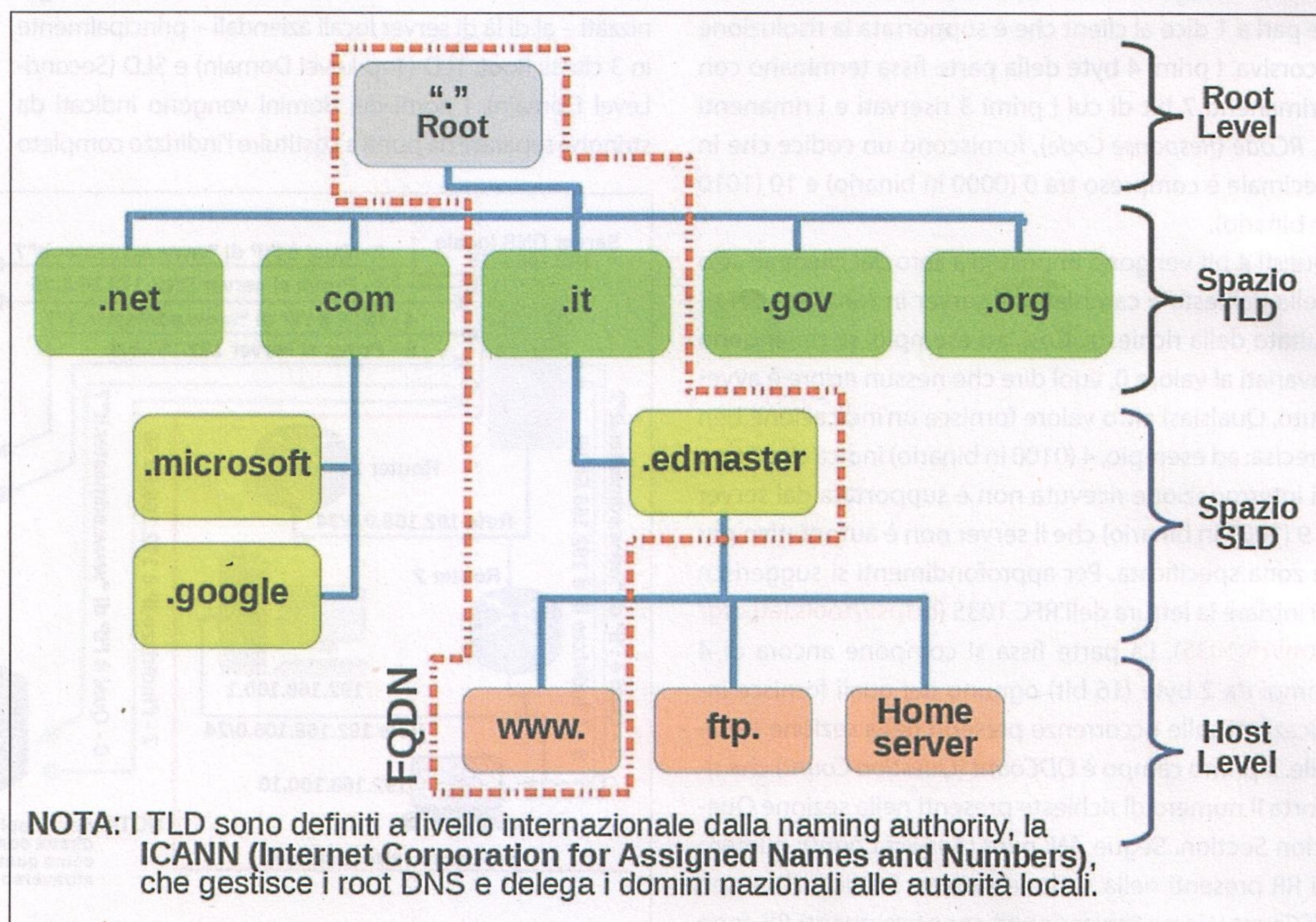



Fig. 3: Struttura ad albero dei nomi a dominio

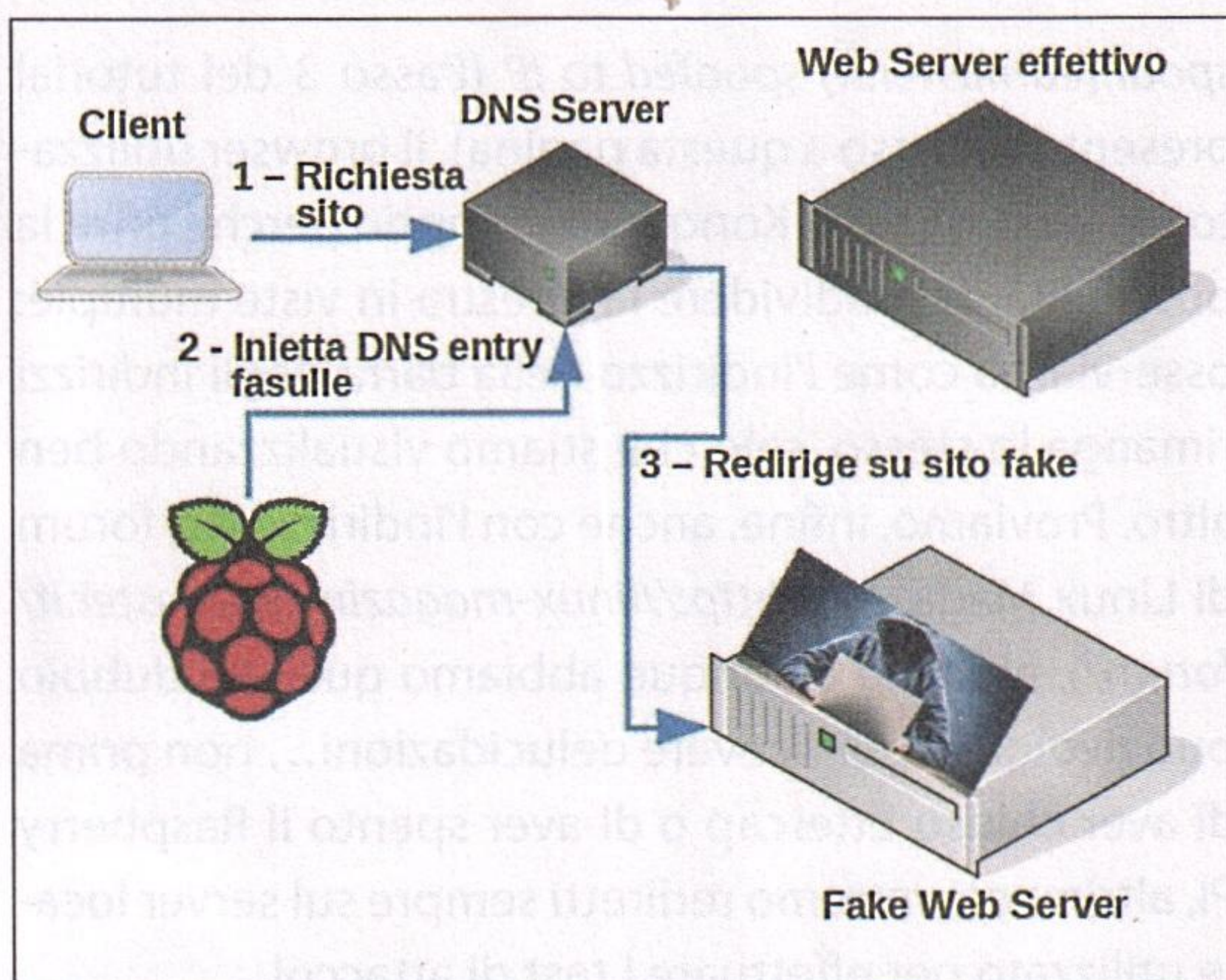



Fig. 4: Schema di principio dell'attacco DNS spoofing

di Internet, è soggetto ad alcuni problemi di sicurezza originati, in buona sostanza, dal fatto che tutta l'informazione fornita dal servizio non necessita di alcun tipo di autenticazione. Ponendoci nel mezzo tra un computer client e il suo server DNS possiamo mettere in pratica un attacco che va sotto il nome di DNS spoofing (contraffazione), noto anche come DNS Cache Poisoning (Fig. 4). In queste pagine ci occuperemo di questo tipo di attacco ipotizzando di essere collegati alla stessa LAN della macchina vittima visto che in buona sostanza l'attacco è basato è un ARP Poisoning (ne abbiamo già parlato su Linux Magazine 176 - Luglio 2017). Premettiamo questo scenario poiché per effettuare lo stesso tipo di attacco ci si potrebbe trovare anche all'esterno della LAN: in questo caso andrebbe attaccato direttamente il server DNS della vittima, fornendogli informazioni "alterate", ovvero spacciandosi per un server DNS di livello superiore che poi quest'ultimo passerebbe legittimamente alla vittima. Questo attacco è noto come DNS Rebinding. Il DNS Spoofing è un attacco che vede un host senza autorità gestire un DNS e tutte le sue richieste, ovvero nel rispondere, alle query DNS della vittima spacciandosi per il suo server DNS, con informazioni false in modo da dirottare la vittima su siti o server fasulli. Questa tecnica non sarebbe possibile se non si fosse collegati nel mezzo tra la vittima e il server DNS, proprio perché abbiamo bisogno di ricevere le query DNS della vittima e rispondergli prima del vero DNS.

A differenza del phishing, la vittima, pur digitando l'indirizzo corretto del sito che vuole visitare e vedendolo visualizzato sul proprio browser, può venire dirottato su un sito fasullo in tutto e per tutto identico all'originale e questo con il solo scopo di rubare i dati sensibili digitati dagli ignari utenti. È un attacco molto pericoloso in quanto di non facile rilevazione: mostreremo come individuare un attacco di questo tipo.

TUTTO L'OCCORRENTE

Faremo uso di Ettercap (https://ettercap.github.io/ettercap/) per simulare un attacco e di Wireshark (www.wireshark.org) per rilevarlo. Come sistema attaccante utilizzeremo una Raspberry Pi dove installeremo Ettercap e il Web server Apache (*sudo apt-get install apache2*, per il nostro test non ci occorre l'interprete PHP), ma qualunque PC va bene. Il Web server Apache lo utilizzeremo per dimostrare il redirect sul falso sito pertanto assicuriamoci che sia attivo prima di iniziare la prova, ad esempio collegandoci all'indirizzo *http://127.0.0.1* con un qualsiasi browser installato su Raspberry Pi. A questo punto, seguiamo i passi del tutorial. Il risultato vede un redirect, ma il sito usato per i nostri test (*www.edmaster.it*) è sempre lì, nessuno l'ha toccato! Semplicemente il Raspberry Pi ha intercettato l'indirizzo e ha rediretto l'ignaro utente sul proprio server Web locale.

Fig. 5: DNS Spoofing catturato da Wireshark

NOTA

RICORSIVA O ITERATIVA

Chi effettua la ricerca?

Una richiesta iterativa è visibile in fig. 2: il server contattato non trova la risposta al posto del client ma piuttosto fornisce a quest'ultimo (server locale aziendale o del provider) le coordinate per risolvere lo step successivo fino all'ottenimento della risoluzione. Il lavoro di ricerca è pertanto svolto dal DNS locale (aziendale o del provider). Una richiesta ricorsiva (bit flag RD a 1) affida il compito di tradurre il nome ai server che vengono contattati. Così il DNS locale chiede al DNS root il quale fa la richiesta al server autoritativo dei TLD fino ad arrivare al server autoritativo del dominio. A questo punto, la risposta risalirà i vari server fino a raggiungere il server locale che fornirà la risposta al client iniziale.

Lanciamo Wireshark sul client (**fig. 5**). All'inizio si hanno le consuete false risposte ARP per poi vedersi arrivare le query DNS per il sito di Linux Magazine. Il router nelle query DNS indica l'IP del Raspberry Pi per la risoluzione del nome (RR di tipo SOA) in indirizzo IP. A questo punto la vittima, senza più tener conto delle vere risposte DNS che gli arriveranno, inizia la sessione TCP con il Raspberry Pi credendo di aver ricevuto l'indirizzo *www.edmaster.it* o suo sotto-dominio.

Che lo spoofing abbia avuto successo è confermato anche dai messaggi nella schermata di shell di Ettercap dove appaiono degli eloquenti *dns_spoof:[edmaster.it] spoofed to IP* (Passo 3 del tutorial presente in basso a questa pagina). Il browser utilizzato nel nostro test è Konqueror, proprio perché offre la possibilità di suddividere la finestra in viste multiple: osserviamo come l'indirizzo nella barra degli indirizzi rimanga lo stesso, solo che stiamo visualizzando ben altro. Proviamo, infine, anche con l'indirizzo del forum di Linux Magazine (*http://linux-magazine.edmaster.it/forum/*), al quale chiunque abbiamo qualche dubbio può rivolgersi per ricevere delucidazioni… non prima di aver chiuso Ettercap o di aver spento il Raspberry Pi, altrimenti verremo rediretti sempre sul server locale utilizzato per effettuare i test di attacco!

NOTA

DNS REBINDING

Di cosa si tratta?

È una tecnica di attacco che può sfruttare metodi differenti, ma tutti con il medesimo obiettivo finale: spingere l'utente a visitare un sito malevolo. Il DNS Rebinding è piuttosto difficile da attuare, per questo motivo abbiamo optato nella dimostrazione per un DNS Spoofing. Infatti, per la dimostrazione necessiteremo di un PC che funzioni da nameserver con un proprio dominio registrato e i RR di tipo NS devono puntare ad una macchina sotto il nostro controllo che presenti un IP pubblico. A tutto ciò va aggiunto l'uso del software rebind (https://code.google.com/archive/p/rebind/).

L'ATTACCO IN PRATICA

SEMPLICI PASSI PER PORTARE A TERMINE IL NOSTRO TEST

1 DOMINIO E IP

Dal Raspberry Pi lanciamo **sudo nano /etc/ettercap/etter.dns.** Aggiungiamo, in un punto qualsiasi, la riga **edmaster.it A IP** seguita da ***.edmaster.it A IP** e da www.edmaster.it **PTR IP** laddove **A** e **PTR** sono tipi di record DNS. In luogo di **IP** inseriremo l'indirizzo di rete del Raspberry Pi (**ifconfig**): 192.168.0.3, nel nostro caso.

2 SCANSIONE

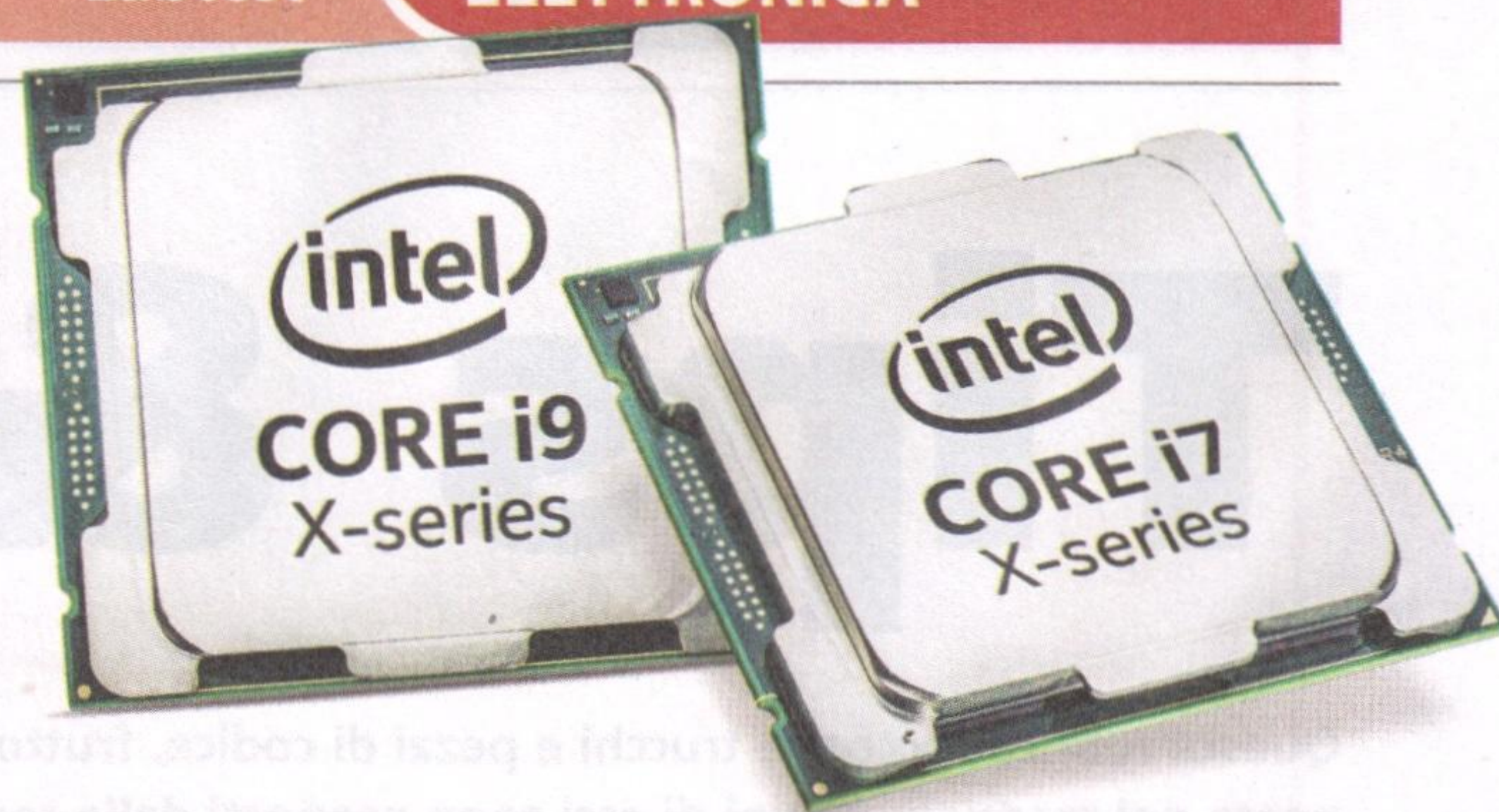
Salviamo le modifiche, lanciamo **Ettercap:** dal menu **Sniff** optiamo per **Unified Sniffing.** Nella pop-up **ettercap Input** selezioniamo la scheda di rete (di solito **eth0**), quindi clicchiamo su **OK.** Da **Hosts**, scegliamo **Scan for Hosts.** Al termine della scansione, optiamo per **Hosts list** (menu **Hosts**). Selezioniamo l'IP della vittima e clicchiamo su **Add to Target 1.**

3 PLUGIN

Selezioniamo l'IP del Router/Gateway (impartiamo **route -n** se non sappiamo quale sia) e clicchiamo su **Add to Target 2.** Clicchiamo su **MiTM**, quindi su **ARP poisoning.** Nella pop-up **MITM Attack: ARP Poisoning** spuntiamo **Sniff remote connections.** Confermiamo con **OK.** Ora, dal menu **Plugins** optiamo per **Manage the plugins.**

4 CONTRAFFAZIONE

Troviamo il plugin **dns_spoof** e abilitiamolo cliccando due volte sul nome stesso: apparirà un asterisco ad indicarne l'abilitazione. Da **Start**, clicchiamo su **Start sniffing.** Lanciamo un browser sul client "vittima" e colleghiamoci a www.edmaster.it. Sorpresa! Appare la schermata di benvenuto di Apache installato sul Raspberry Pi!

INTEL COLPISCE ANCORA

INTEL NON HA POTUTO FARE ALTRO CHE STARE A GUARDARE CON UN CERTO NERVOSISMO LA RINASCITA DI AMD GRAZIE AI PROCESSORI "RYZEN". IL LEADER DEL MERCATO REPLICA ORA PRESENTANDO LE STRAORDINARIE CPU "SKYLAKE X" E "KABY LAKE X". IOPROGRAMMO LE HA GIÀ TESTATE

La presentazione del formidabile processore "Ryzen" R7 1800X da parte di AMD ha creato un grande scalpore e, nelle prove di velocità, l'R7 si è piazzato pericolosamente alle calcagna della CPU top di gamma di Intel, il Core i7-6950X. Con il Core i9-7900X (Skylake X; 1050 Euro) e il Core i7-7740X (Kaby Lake X; 360 Euro), Intel punta ora a rimettere in riga l'avversario. Riuscirà a raggiungere l'obiettivo?

COSA HANNO IN COMUNE

Entrambi i processori si rivelano già costosi, ma incombono anche spese extra, dato che Intel ha corredato le due CPU di un nuovo socket. Una nuova mainboard con socket LGA-2066, come l'Asus Strix X299 (350 Euro) utilizzata per questo test, richiederà una nuova RAM (nel test è stata impiegata la Corsair Vengeance DDR4-3000 - 4 x 8 Gigabyte; 300 Euro), nonché una nuova ventola (a partire da 50 Euro). Inoltre per entrambe le CPU Intel ha rinunciato ad un chip grafico integrato e per la riproduzione delle immagini si rende quindi d'obbligo una scheda grafica separata. Tutto questo non si rende necessario con le CPU Ryzen e Core i7-6950X.

CORE I9-7900X CON DIECI CORE

Analogamente al vecchio processore Core i7-6950X, anche il nuovo Core i9-7900X ostenta la propria superiorità con i suoi dieci core, che dovrebbero offrire una frequenza di clock leggermente più elevata: 3,3 anziché 3,0 Gigahertz in modalità normale, arrivando a 4,3 anziché 3,5 Gigahertz, sfruttando la tecnologia turbo. Il Core i7-7740X offre una frequenza di clock ancora più elevata: 4,3 Gigahertz in modalità normale e addirittura 4,5 GHz in modalità turbo. La i7-7740X offre però solo quattro core su cui poter distribuire i dati.

I nuovi processori si sono rivelati però anche un po' deludenti: infatti il Core i7-7740X ha offerto solo la stessa velocità di lavoro del più economico Core i7-7700K (350 Euro). Il Core i9-7900X a dieci core, si è invece subito aggiudicato la vittoria per la velocità. Fino ad oggi, nei laboratori di ioProgrammo, nessun altro processore ha offerto una velocità di lavoro così elevata (comparativa in basso a sinistra). Unico neo riscontrato è il fabbisogno energetico che, rispetto al Core i7-6950X, si è rivelato decisamente più elevato. Con il Core i9-7900X utilizzato a pieno carico, il PC impiegato per il test ha richiesto 48 Watt in più di corrente (393 anziché 345 Watt). Il Core i7-7740X vanta invece un consumo energetico più contenuto, dato che per il PC sono stati necessari 340 Watt, mentre sul Core i7-7700K è stato riscontrato un consumo di 333 Watt.

I PROCESSORI PIÙ VELOCI

Il processore Core i9-7900X della serie Skylake X di Intel si piazza al primo posto, con un notevole distacco su tutte le altre CPU.

I RISULTATI IN BREVE	INTEL CORE i9-7900X Prezzo: 1.050 Euro **Serie/Socket:** Skylake X / LGA 2066 **Frequenza di clock della CPU:** 3,3 GHz **Numero dei core:** 10 **Processore grafico:** non presente		INTEL CORE i7-7740X Prezzo: 360 Euro **Serie/Socket:** Kaby Lake X / LGA 2066 **Frequenza di clock della CPU:** 4,3 GHz **Numero dei core:** 4 **Processore grafico:** non presente	
A quale velocità posso lavorare con la CPU?	Straordinariamente potente: il Core i9 sale al primo posto tra le CPU più veloci.	10,00	Anche il modello Kaby-Lake X Core i7-7740X si rivela piuttosto veloce	8,56
Quanto è veloce il processore nell'elaborazione di calcoli complessi?	In questa prova, il Core i9 vince sull'AMD 1800X, rivelandosi la più veloce CPU.	10,00	Per queste operazioni, la 7740X si rivela più lenta del 68% rispetto al Core i9	6,7
Quanto è veloce il processore con i giochi, con GPU separata GTX 980Ti ?	Con giochi Full-HD: 84 fps Con giochi 4K: 38 fps	8,96	Giochi Full-HD: 84 fps Giochi 4K: 38 fps	8,60
Quanto è veloce con chip grafico integrato?	Nessun chip grafico integrato	0,00	Nessun chip grafico integrato	0,00
Quanto è elevato il consumo energetico [1]?	A pieno carico: molto elevato (393 Watt)* Office: un po' elevato (84 Watt)	4,00	A pieno carico: minimo (340 Watt)* Office: minimo (45 Watt)*	8,40
RISULTATO DEL TEST	buono 8,24		buono 7,20	

[1] Consumo energetico del PC utilizzato per il test

CONCLUSIONI

Sorprendente che nel raffronto delle velocità, il Core i9-7900X si sia piazzato subito al primo posto! Anche il Core i7-7740X si rivela piuttosto veloce, ma non così rapido come il più economico Core i7-7700K. Intel non potrà però rimanere tranquilla, dato che AMD ha lanciato sul mercato il processore "Threadripper" a 16 core, che dovrebbe aggiudicarsi la corona di CPU più veloce. Entro ottobre però, Intel presenterà la sua arma più potente: il Core i9-7980XE dotato di 18 core. La sfida sta diventando avvincente!

Tips & Tricks

Questa rubrica raccoglie trucchi e pezzi di codice, frutto dell'esperienza di chi programma, che solitamente non trovano posto nei manuali. Alcuni di essi sono proposti dalla redazione, o provengono da una ricerca su Internet, altri ancora giungono dai lettori. Chi vuole contribuire, può inviare i suoi Tips & Tricks preferiti. Una volta selezionati, saranno pubblicati nella rubrica. Tutti i tips elencati, ed altri ancora, sono presenti anche nel CD-Rom allegato alla rivista

JAVA

UN CONVERTITORE DI TEMPERATURA

```
import javax.swing.JFrame;
import javax.swing.JOptionPane;
public class TemperatureConvertor {
  public static void main(String args[])
  {
    JFrame answerbox = new JFrame();

    String choice;
    String DEGREE  = "\u00b0";
    double celsius;
    double fahrenheit;

    choice = JOptionPane.showInputDialog("Type 'A' if you want to
             convert Celsius to Fahrenheit\nType 'B' if you want to convert
                                       Fahrenheit to Celsius");

    if(choice.equals("A") || (choice.equals("a")))
    {
      celsius = Double.parseDouble(JOptionPane.showInputDialog(
          "What is the Celsius temperature you want to convert?"));
                               celsius = (celsius * 1.8) + 32;
      JOptionPane.showMessageDialog(answerbox, "Your new
                   temperature is " + celsius + DEGREE + "F");
    }
    if(choice.equals("B") || (choice.equals("b")))
    {
      fahrenheit = Double.parseDouble(JOptionPane.showInputDialog(
         "What is the Fahrenheit temperature you want to convert?"));
      fahrenheit = (fahrenheit - 32) * 5/9;
      JOptionPane.showMessageDialog(answerbox, "Your new
                   temperature is " + fahrenheit + DEGREE + "C");
    }
  }
}
```

GENERAZIONE DI PDF CON ITEXT

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.OutputStream;
import java.util.Date;

import com.lowagie.text.Document;
import com.lowagie.text.Paragraph;
import com.lowagie.text.pdf.PdfWriter;

public class GeneratePDF {

  public static void main(String[] args) {
    try {
      OutputStream file = new FileOutputStream(new File(
                                                "C:\\Test.pdf"));

        Document document = new Document();
        PdfWriter.getInstance(document, file);
        document.open();
        document.add(new Paragraph("Hello Kiran"));
        document.add(new Paragraph(new Date().toString()));

        document.close();
        file.close();

      } catch (Exception e) {

        e.printStackTrace();
      }
    }
}
```

PHP

CALCOLARE LA DIMENSIONE DI UNA DIRECTORY IN PHP

```
<?php
function CalcDirectorySize($DirectoryPath) {
    // I reccomend using a normalize_path function here
    // to make sure $DirectoryPath contains an ending slash
    // To display a good looking size you can use a readable_filesize
    // function.
    $Size = 0;
```

```
    $Dir = opendir($DirectoryPath);
    if (!$Dir) return -1;
    while (($File = readdir($Dir)) !== false) {
        // Skip file pointers
        if ($File[0] == ) continue;
        // Go recursive down, or add the file size
        if (is_dir($DirectoryPath . $File)) {
            $Size += CalcDirectorySize($DirectoryPath . $File .
                                        DIRECTORY_SEPARATOR);
        }
        else {
            $Size += filesize($DirectoryPath . $File);
        }
    }
    closedir($Dir);
    return $Size;
}
?>
```

TRASFORMARE UN ARRAY IN UN CSV

```
function array2csv(array &$array)
{
    if (count($array) == 0) {
        return null;
    }
    ob_start();
    $df = fopen("php://output", 'w');
    fputcsv($df, array_keys(reset($array)));
    foreach ($array as $row) {
        fputcsv($df, $row);
    }
    fclose($df);
    return ob_get_clean();
}
```

JAVASCRIPT

UN HEADER CHE NON VA VIA

```
//Fixed Menu (Header or footer) v1.3.1
//Replace ID below with your own box ID
var boxToAppend = '#lp-pom-box-177';
//Set to 'header' or 'footer'
var headerOrFooter = 'header';

var backgroundCSS = {"position":"fixed", "left":"0", "top":"0px",
                     "bottom":"auto", "width":"100%", "z-index":"899"};
var colorOverlayCSS = {"position":"fixed", "left":"0", "top":"0px",
                       "bottom":"auto", "width":"100%", "z-index":"auto",
                       "border-style":"none none none none"};
var childrenCSS = {"position":"fixed", "left":"auto", "top":"0px",
    "bottom":"auto", "width":"100%", "z-index":"999", "border-style":"none
                   none none none", "border-width":"0px", "background":"none"};

if (headerOrFooter === 'footer') {
    backgroundCSS["top"] = 'auto';
    backgroundCSS["bottom"] = '0px';
    colorOverlayCSS["top"] = 'auto';
    colorOverlayCSS["bottom"] = '0px';
    childrenCSS["top"] = 'auto';
    childrenCSS["bottom"] = '0px';
}
var boxParent = $(boxToAppend).parent();
var boxClone = $(boxToAppend).clone()
boxClone.appendTo(boxParent).css(backgroundCSS).children().remove();
$(boxToAppend).css(childrenCSS);
$(boxToAppend + '-color-overlay').appendTo(boxClone).
                                              css(colorOverlayCSS);
```

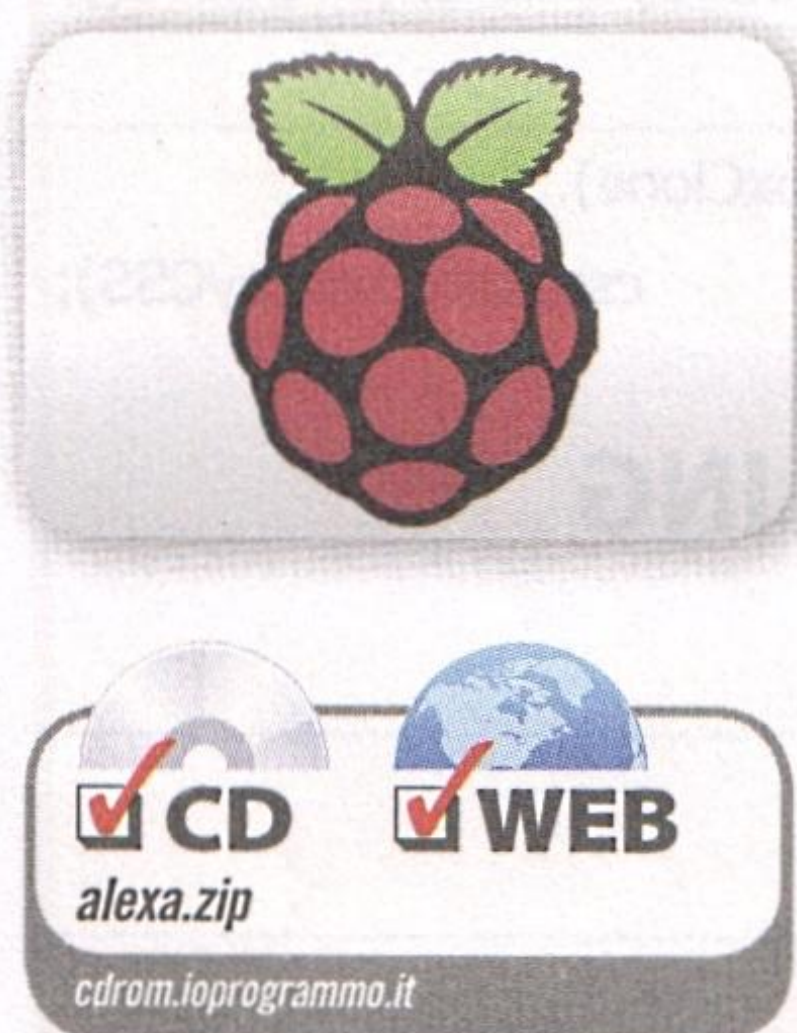
ADDOLCIAMO LO SCROLLING

```
lp.jQuery(function($) {
    // The speed of the scroll in milliseconds
    var speed = 1000;

    // Find links that are #anchors and scroll to them
    $('a[href^=#]')
      .not('.lp-pom-form .lp-pom-button')
      .unbind('click.smoothScroll')
      .bind('click.smoothScroll', function(event) {
        event.preventDefault();
        $('html, body').animate({ scrollTop: $( $(this).attr('href')
                                   ).offset().top }, speed);
      });
  });
```

BLOCCARE IL CLICK CON IL TASTO DESTROY NEL BROWSER

```
<script type="text/javascript">
function f1() {
  if(document.all) { return false; }
}
function f2(e) {
  if(document.layers || (document.getElementById && document.all)) {
    if(e.which==2 || e.which==3) { return false; }
  }
}
if(document.layers) {
  document.captureEvents(Event.MOUSEDOWN);
  document.onmousedown = f1;
}
else {
  document.onmouseup = f2;
  document.oncontextmenu = f1;
}
document.oncontextmenu = new function("return false");
```

ALEXA: ECCO COME PROGRAMMARLA!

ALEXA, L'ASSISTENTE VOCALE DI AMAZON È QUANTO PIÙ DI VICINO CI SIA AD UN VERO ASSISTENTE VIRTUALE. IMPARIAMO A CONOSCERLA, UTILIZZARLA E A PROGRAMMARE UNA SKILL PER ALEXA CON LE AMAZON LAMBDA

Da un paio d'anni si sono affacciati sul mercato gli assistenti vocali "da tavolo". Da tempo presenti sui nostri cellulari (pensate ad "Ok Google" o a Siri), gli assistenti vocali stanno entrando in casa nostra grazie ai colossi del Web, in particolare Google con il suo Home e Amazon con la famiglia di prodotti Echo stanno invadendo le case dei mercati su cui sono stati lanciati. Ovviamente Apple non sta a guardare e ha da poco lanciato una versione "boxed" di Siri, chiamata HomePod. Echo, la cui tecnologia è nata nello sviluppo dello smartphone di Amazon, Fire, considerato un fallimento e ritirato dal mercato, è stato annunciato nel novembre 2014 per il mercato americano. Da allora ne sono stati venduti 7 milioni di esemplari. Amazon Echo è tra i primi prodotti dell'Internet of things che inizia ad avere una considerevole diffusione nelle case, anche grazie alle skills sviluppate da terze parti. A settembre 2016 Echo è arrivato in Gran Bretagna e in Germania.

Nella visione di Amazon, lo scambio vocale garantisce una modalità di interazione con i servizi "senza alcuna frizione". Non serve guardare uno schermo, si chiama semplicemente "Alexa" e le si chiede di ordinare una pizza, un'auto Uber o le informazioni sul traffico per andare in ufficio. Nel corso di questo articolo impareremo a programmare delle Skill per Amazon Echo, utilizzando gli SDK e i servizi di Amazon.

Conoscenze richieste

Programmazione a oggetti, Amazon Lambda

Software

Un terminale, un editor di testo, account amazon AWS (gratuito), Amazon Echo o Raspberry Pi per parlare con Alexa.

Impegno

Tempo di realizzazione

A COSA SERVE UN ASSISTENTE VOCALE E DOVE POSSO TROVARNE UNO?

OK, cos'è quindi un assistente vocale? Un assistente vocale (da questo momento ci focalizzeremo su Alexa) è un device (che può essere dedicato o uno smartphone) il cui software ascolta i nostri comandi e agisce di conseguenza, grazie all'intelligenza artificiale e al riconoscimento della voce.

Prendiamo appunto Alexa. Possiamo chiedere ad Alexa (in inglese, purtroppo l'italiano non è ancora supportato, ma lo sarà presto) di farci ascoltare della musica specifica, di impostare una sveglia, di leggerci le news o le previsioni meteo per una determinata località. Grazie alle

Fig. 1: La famiglia Amazon Echo

integrazioni con API di terze parti è possibile espandere Alexa facendole gestire per noi le luci di casa, la temperatura del nostro termostato smart e molto altro. È anche possibile chiedere ad Alexa cose più "frivole" come di raccontarci una barzelletta o una storia della buonanotte. Ecco alcune delle innumerevoli domande (o richieste) che possiamo fare ad Alexa (in inglese):

- Alexa, wake me up at 7 in the morning
- Alexa, what's on my calendar for tomorrow?
- Alexa, what's in the news?
- Alexa, turn on the lights
- Alexa, tell me a joke

Come vedete è possibile chiedere quasi qualsiasi cosa, a seconda delle skill installate sul nostro account. OK, ma cos'è una skill? Una skill, nell'ecosistema Amazon Alexa, è una "capacità" che possiamo aggiungere ad Alexa. Ad esempio una skill da aggiungere potrebbe essere quella di leggerci l'oroscopo del giorno.

Una volta installata la skill tramite l'apposito store di Amazon.com, sarà subito disponibile per il nostro device. Guardate alle skill un po' come alle app del vostro smartphone. Uno smartphone da solo fa "poco", ma con le app può fare praticamente tutto. La stessa cosa vale per Alexa e le sue skill. Per questo nel corso dell'articolo svilupperemo una skill tutta nostra.Tutto chiaro? Passiamo dal "cosa" al "come". La parte più difficile è recuperare uno di questi device. In Italia non sono ufficialmente in vendita, quindi le soluzioni sono le seguenti:

- Potete organizzarvi con un amico o un parente residente in uno dei paesi dove Amazon Echo è in vendita e che sia disposto a spediverlo
- Potete organizzare quel week-end a Londra a cui pensavate da un po' e dedicarvi allo shopping
- Potete comprare il device che preferite su eBay o altre piattaforme
- Potete installare il software di Amazon Echo su un Raspberry Pi o su un computer qualsiasi, di più facile reperibilità. Vedremo come fare nel prossimo paragrafo.

Indipendentemente dal possesso di un vero device Echo o di un Raspberry Pi modificato, il codice che andremo a scrivere sarà lo stesso. Personalmente ho potuto comprare un Amazon Echo Dot durante un viaggio di lavoro negli Stati Uniti dove è possibile trovarlo in qualsiasi negozio di elettronica.

INSTALLARE ALEXA SU UN RASPBERRY PI

Se possedete già un Amazon Echo potete saltare questa parte. Se invece non sapete cosa sia un Raspberry Pi, sappiate che è un single-board computer (un calcolatore implementato su una sola scheda elettronica) sviluppato nel Regno Unito dalla Raspberry Pi Foundation. Il suo lancio al pubblico è avvenuto il 29 febbraio 2012. Finora, ne sono state prodotte otto versioni (Modelli: A, B, A+, B+, 2, Zero, 3, Zero W) con prezzi da 5 a 35 dollari statunitensi. L'idea di base è la realizzazione di un dispositivo economico, concepito per stimolare l'insegnamento di base dell'informatica e della programmazione nelle scuole. Il progetto ruota attorno a un System-on-a-chip (SoC) Broadcom, che incorpora un processore ARM, una GPU VideoCore IV, e 256 o 512 Megabyte o 1 Gigabyte di memoria. Il progetto non prevede né hard disk né una unità a stato solido, affidandosi invece a una scheda SD per il boot e per la memoria non volatile. La scheda è stata progettata per ospitare sistemi operativi basati sul kernel Linux. Possiamo installare, grazie ad Amazon, l'assistente Alexa anche su questo tipo di macchine. In Rete sono disponibili diverse guide, ma cercheremo di rendere questa procedura la più semplice e veloce possibile utilizzando AlexaPi, un client per gli Alexa Web service che trovate qui: https://github.com/alexa-pi/AlexaPi.
Cosa ci serve:

- Un Raspberry Pi (3, preferibilmente, perché già con Wi-Fi e Bluetooth integrato) o un qualunque computer basato su Linux
- Un microfono USB (ne trovate moltissimi per qualche euro online)
- Una card MicroSD da almeno 8 GB
- Un cavo di alimentazione Micro-USB
- Una cassa (da connettere al Raspberry tramite jack audio)
- Una tastiera ed un mouse per il setup iniziale

Ok, pronti? Primo step. Registrare il vostro device gratuitamente all'indirizzo https://developer.amazon.com, accedete col vostro account Amazon o registratevi se non lo avete ancora fatto.

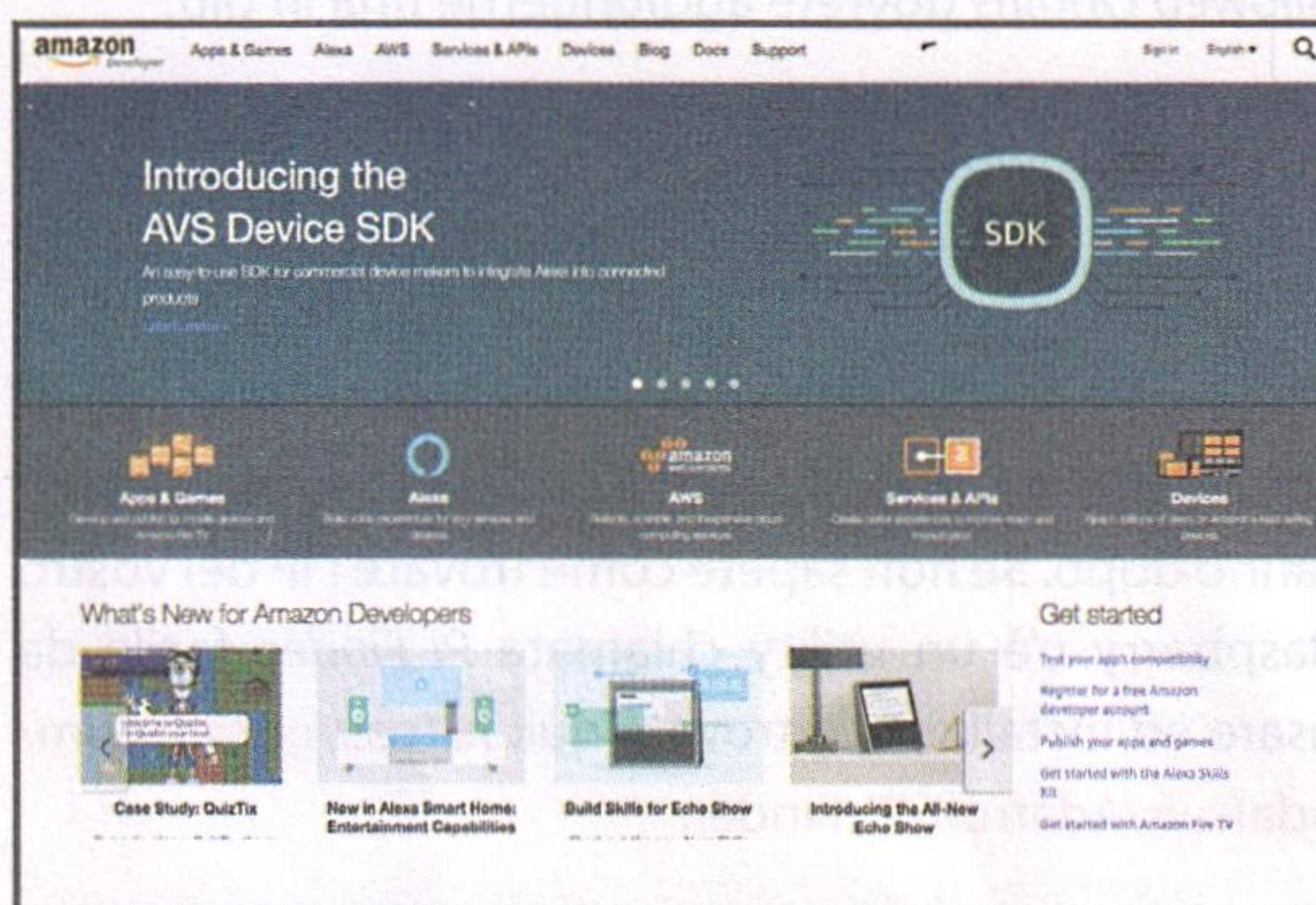
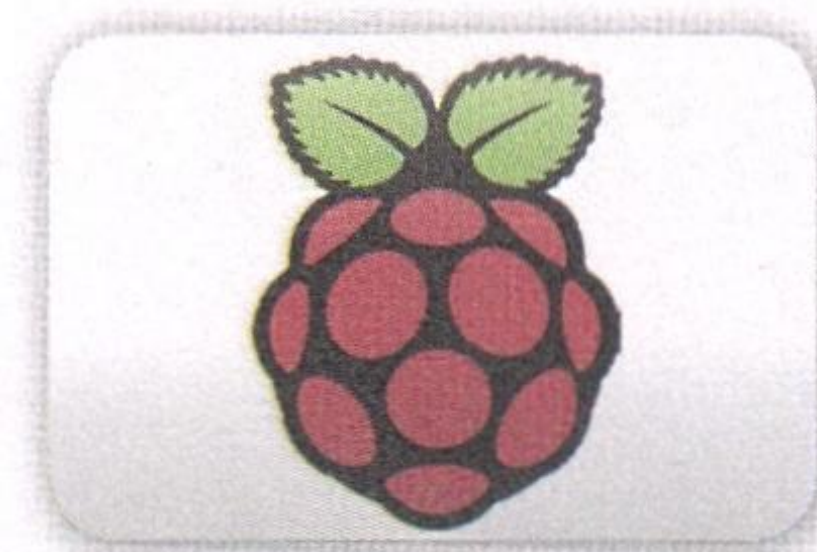



Fig. 2: Il sito developer.amazon.com

Cliccate poi su *Alexa* nel menù in cima alla pagina e sull'icona *Alexa Voice Service*.

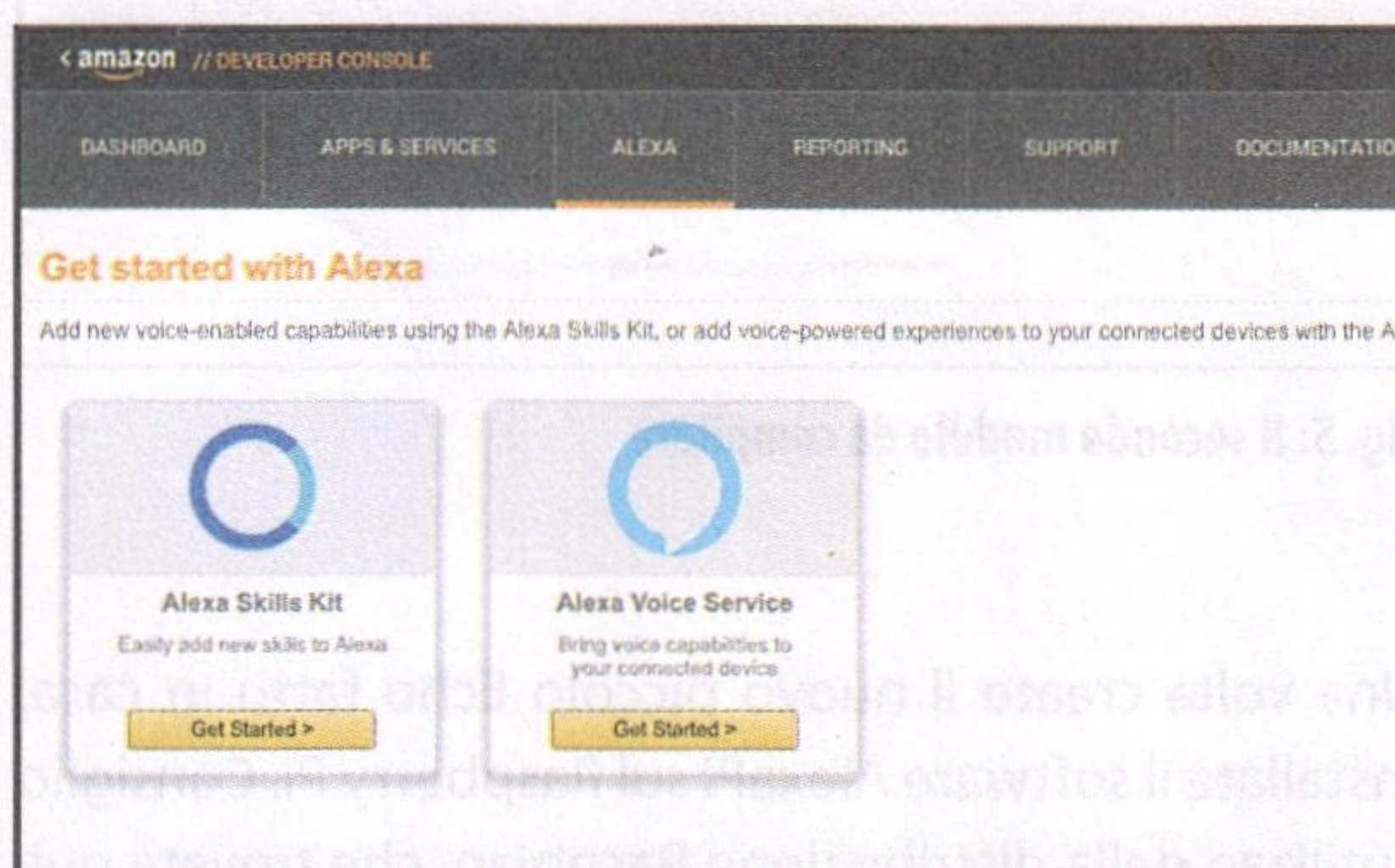



Fig. 3: Cliccate sull'icona "Voice Service"

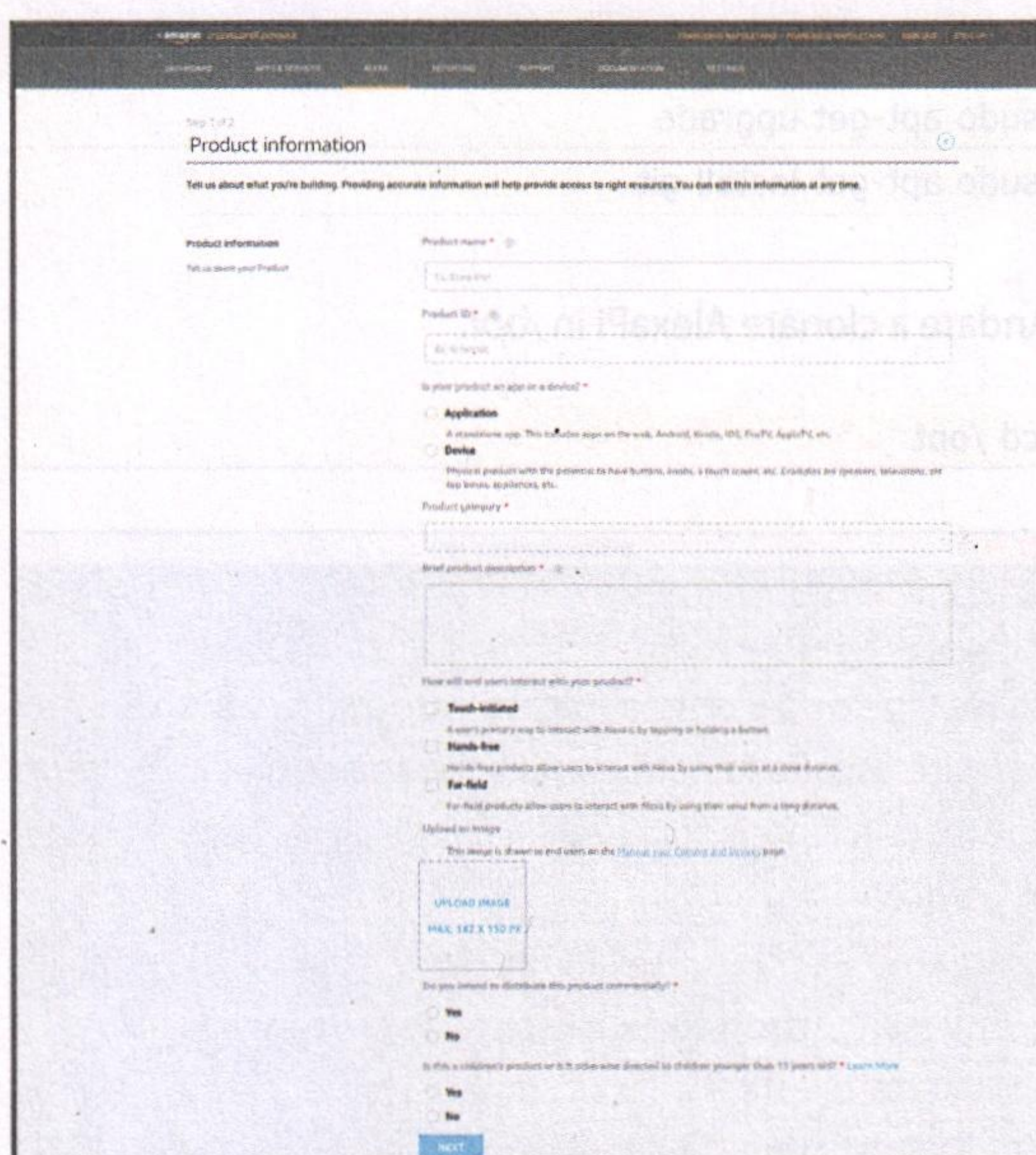
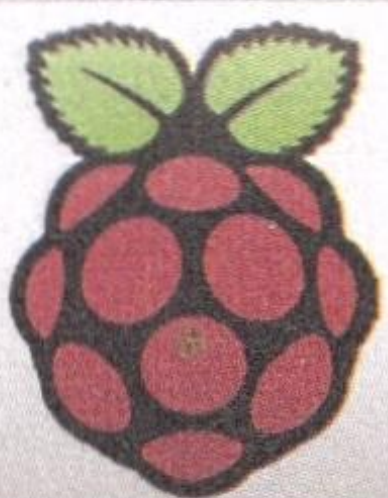



Fig. 4: Il primo modulo da compilare

Compilate il modulo che si aprirà. L'unica accortezza che dovete avere sarà quella di selezionare *Hands-free* quando vi verrà chiesto il tipo di device da creare. Create quindi un nuovo security profile nella schermata successiva. Non appena compare la nuova schermata *Platform Information* compilate il modulo selezionando *Web* come piattaforma e inserite i valori come spiegato di seguito.
Allowed Origins dovrete aggiungerne una in più:

- *http://localhost:5050*
- *http://IP_DEL_VOSTRO_RASPBERRY:5050*

Gli stessi valori andranno inseriti negli *Allowed Return Urls*. Prendete nota di *Client ID* e *Client Secret*. Ci serviranno dopo. Se non sapete come trovare l'IP del vostro Raspberry c'è un utility chiamata *Pi Finder*, facile da usare ed installare. La trovate qui: https://github.com/adafruit/Adafruit-Pi-Finder.

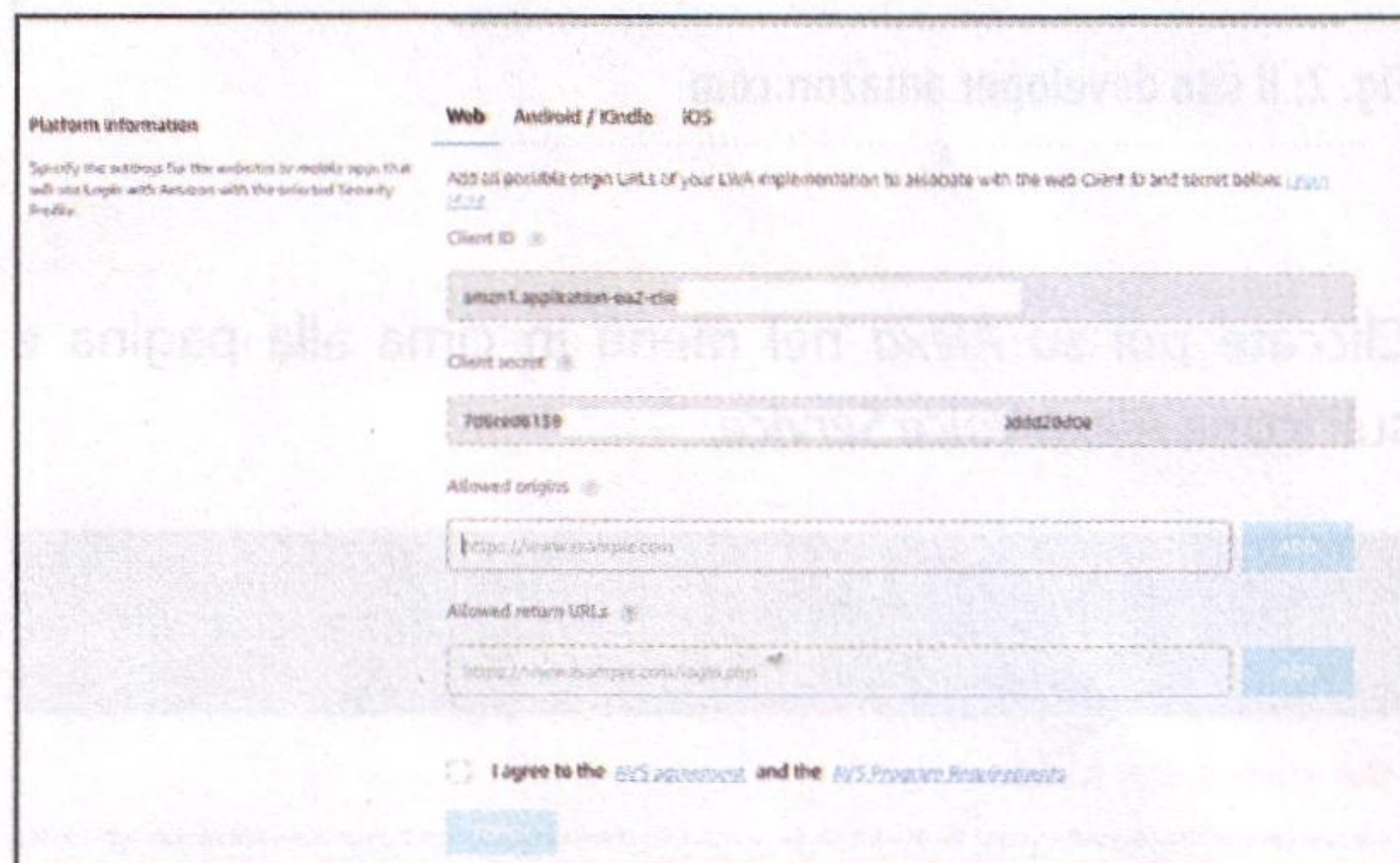



Fig. 5: Il secondo modulo da compilare

Una volta creato il nuovo piccolo Echo fatto in casa, installate il software AlexaPi sul Raspberry Pi. Consiglio l'utilizzo della distribuzione Raspbian, che trovate qui: https://www.raspberrypi.org/downloads/raspbian/. Installate per prima cosa *git*:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install git
```

Andate a clonare AlexaPi in */opt*.

```
cd /opt
```

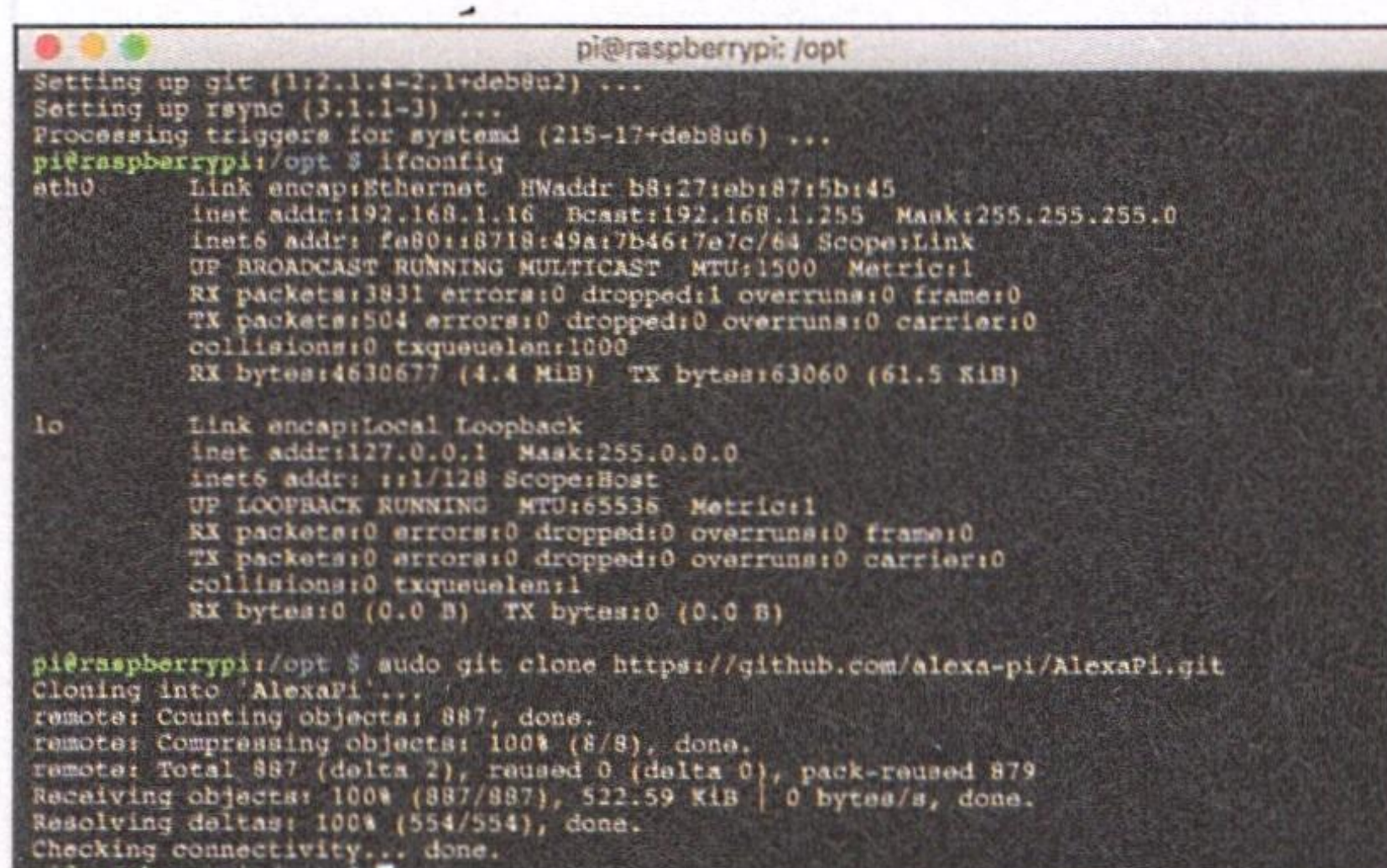



Fig. 6: Download di AlexaPi

```
sudo git clone https://github.com/alexa-pi/AlexaPi.git
```

Sempre rimanendo nella cartella */opt* digitate ora questo comando per lanciare il setup di AlexaPi

```
sudo ./AlexaPi/src/scripts/setup.sh
```

Inserite i dati richiesti e le chiavi provenienti dal pannello *Amazon Developer*. Dovrete autorizzare ora il device nel pannello Amazon, sarà necessario farlo solo la prima volta. Visitate usando un browser *http://IP_DEL_VOSTRO_RASPBERRY:5050* ed eseguite le operazioni richieste.
Lanciate ora il comando

```
sudo systemctl start AlexaPi.service
```

Se tutto è andato per il verso giusto alla domanda "Alexa" riceverete uno "Yes" come risposta. Se ci sono problemi verificate lo stato del servizio digitando

```
sudo systemctl status AlexaPi.service
```

Consultate la documentazione di AlexaPi, è molto ben fatta. Ad ogni reboot del Raspberry Alexa partirà e sarà pronta ad ascoltarvi.

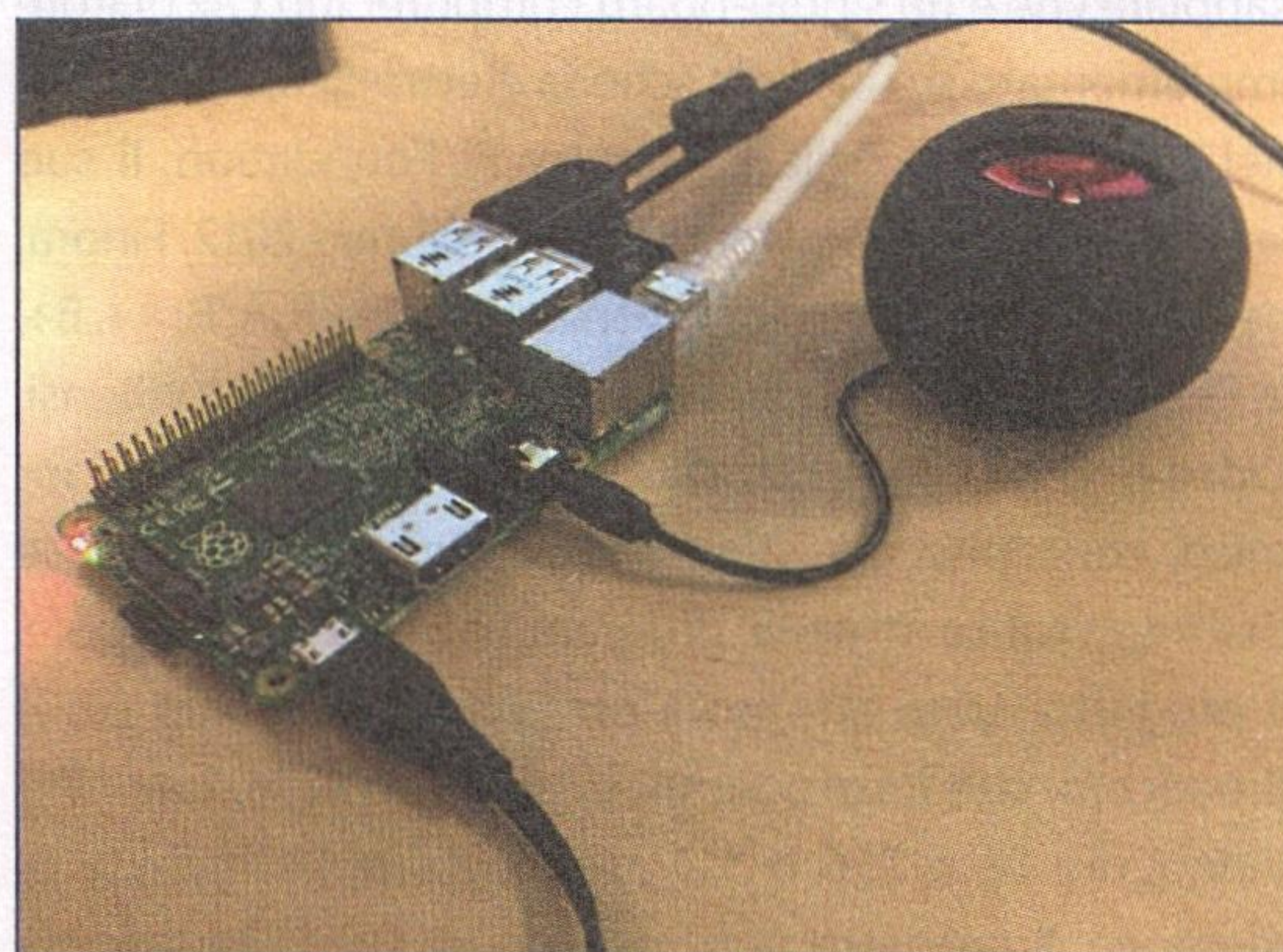



Fig. 7: Un Raspberry Pi collegato ad uno speaker, con installata Alexa.

Vediamo ora come programmare Alexa in modo da poter sfruttare al meglio le capacità dell'assistente vocale di Amazon!

LA NOSTRA PRIMA SKILL ALEXA

Abbiamo adesso un device (qualunque esso sia) collegato ad Alexa. Ovviamente, da bravi programmatori non vogliamo solo usarlo come utenti, ma vogliamo anche poterlo programmare. Esistono due modi per poterlo fare.
Il primo, più semplice e alla portata anche di chi non ha mai programmato in vita propria, è usare

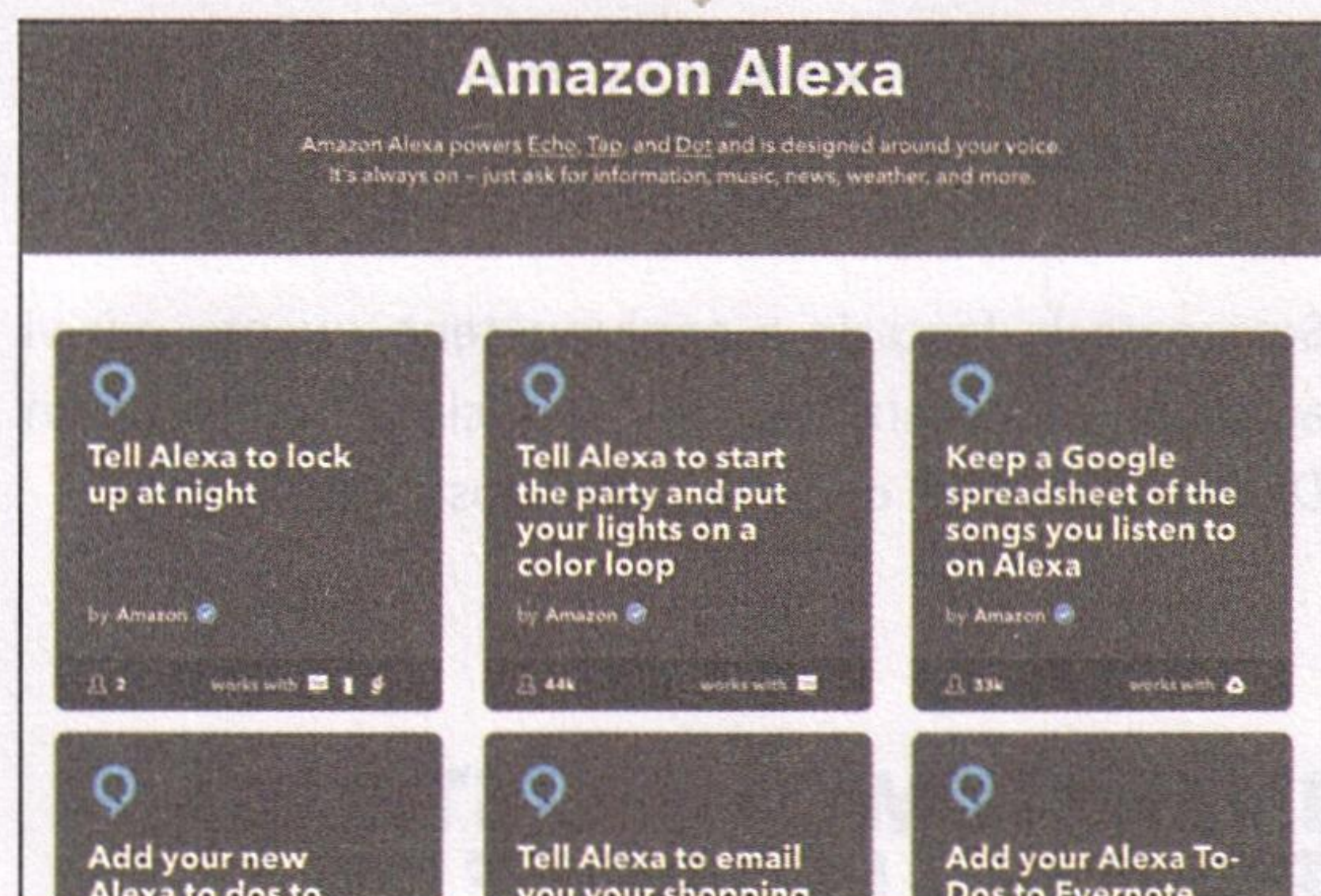



Fig. 8: La pagina su IFTTT.com che raccoglie le migliori applet per Alexa.

IFTTT.com. Il sito ci permetterà di creare delle applet che alla pronuncia di alcuni frasi specifiche eseguiranno delle azioni usando i servizi integrati con IFTTT.com. Niente di più facile, provare per credere: potrete controllare facilmente i vari componenti della vostra smart home o servizi come Google Calendar e Gmail. Il secondo modo per programmare Alexa è più tradizionale. Useremo i servizi Web di Amazon per creare delle Skill che verranno eseguite da Alexa alla pronuncia di specifiche frasi. Se non vi siete registrati su https://developer.amazon.com fatelo ora.

Creeremo una skill che ci permetterà di chiedere ad Alexa alcune domande relative all'informatica e di ricevere una risposta.

Ecco qualche esempio, ovviamente in inglese, visto che l'italiano non è ancora supportato:

- What's the definition of array?
- What's the meaning of MVC?
- Define recursion
- What is a variable?

Utilizzeremo *Node.js* e *Amazon AWS Lambda* per questa nuova Skill.

Ma cos'è una lambda? AWS Lambda è un servizio di elaborazione serverless che esegue codice in risposta a determinati eventi. Il codice che si decide di eseguire su AWS Lambda viene chiamato "funzione Lambda".

Una volta creata, una funzione Lambda è sempre attiva e pronta per essere attivata, in modo analogo a una formula di un foglio di lavoro. Ciascuna funzione include codice e alcune informazioni di configurazione, quali il nome della funzione e i requisiti in termini di risorse. Le funzioni Lambda sono "stateless", prive di affinità con l'infrastruttura sottostante, perciò AWS Lambda può avviare qualsiasi numero di copie della funzione, in base alla frequenza degli eventi in entrata. Nel nostro caso l'evento sarà la chiamata ad Alexa che effettueremo con la voce. Il codice della Lambda verrà eseguito, ritornerà una risposta e Alexa pronuncerà l'output. Facile no?

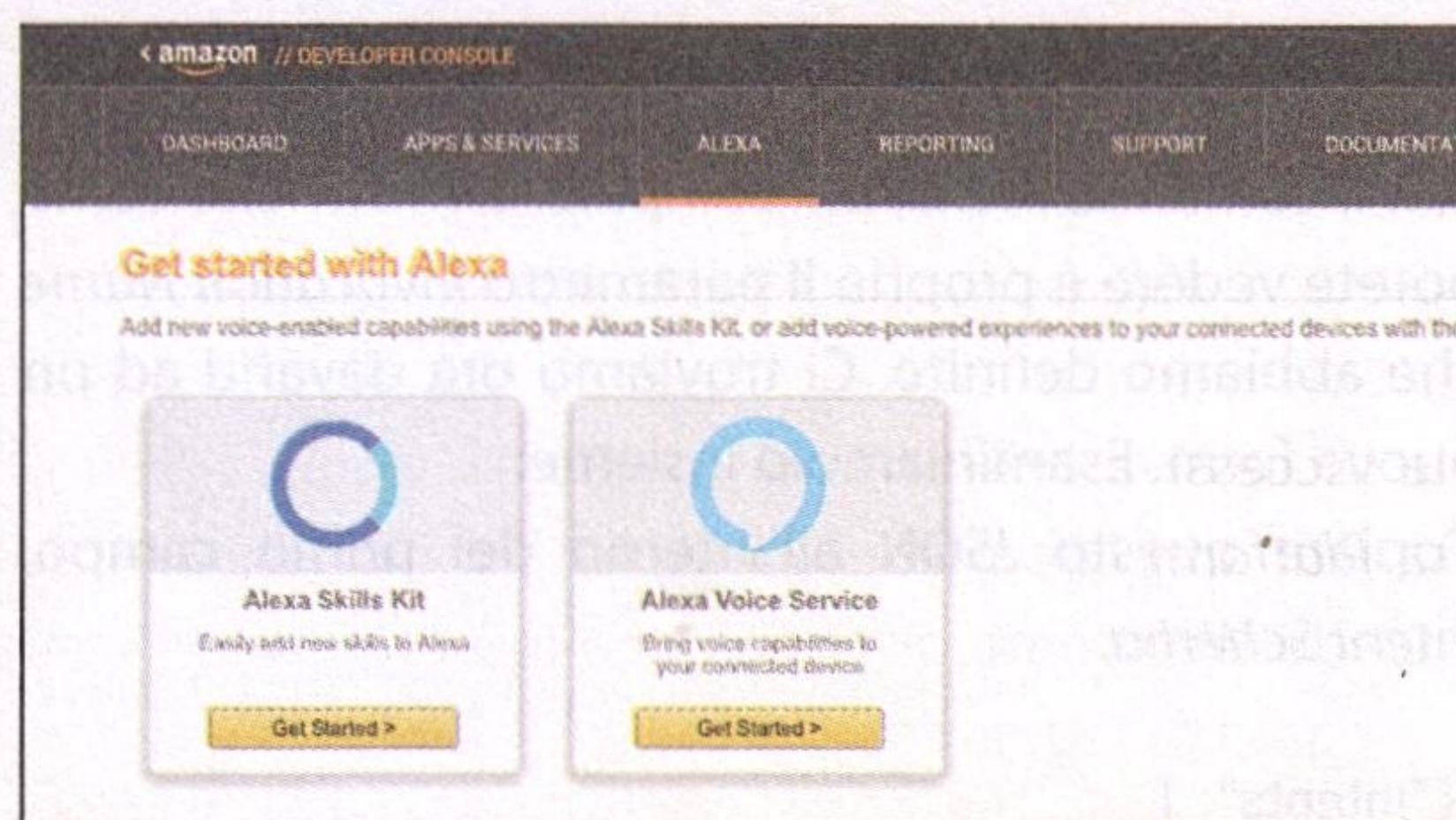



Fig. 9: Cliccate la prima icona

PARTIAMO!

Ok, avete il vostro account su https://developer.amazon.com. Bene. Loggatevi e cliccate su *Alexa* nel menù in alto. Cliccate su *Alexa Skill Kit* e poi su *Add a new skill*.

Compilate il form selezionando questi valori:

- SkillType: *Custom Interaction Model*
- Language: *English (U.S.)*
- Name: *IoProgrammo*
- InvocationName: *computer terms*

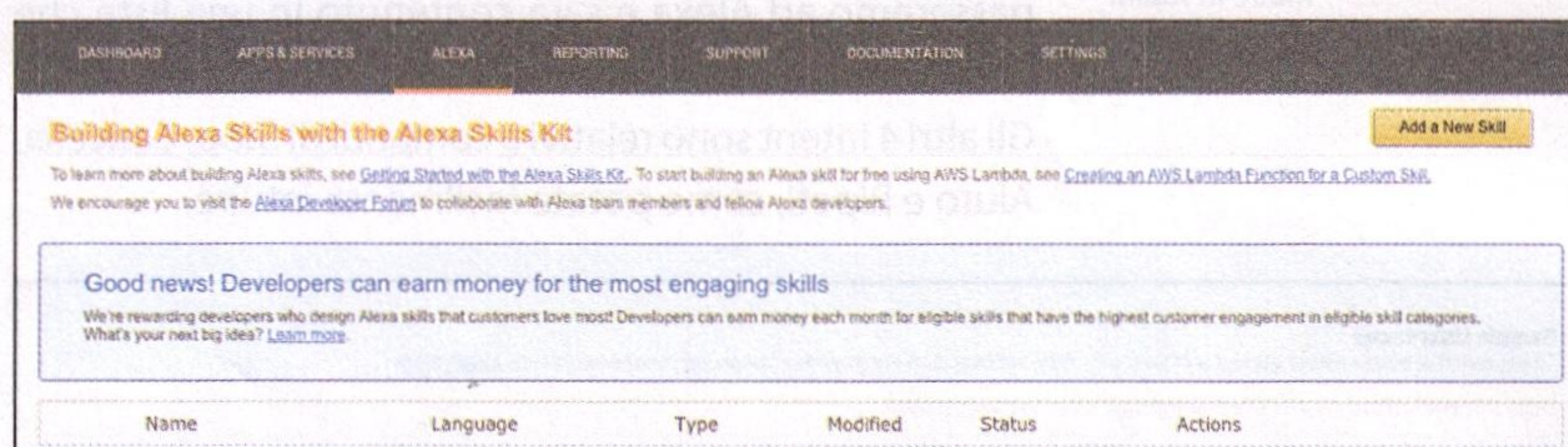



Fig. 10: Aggiungete una nuova skill

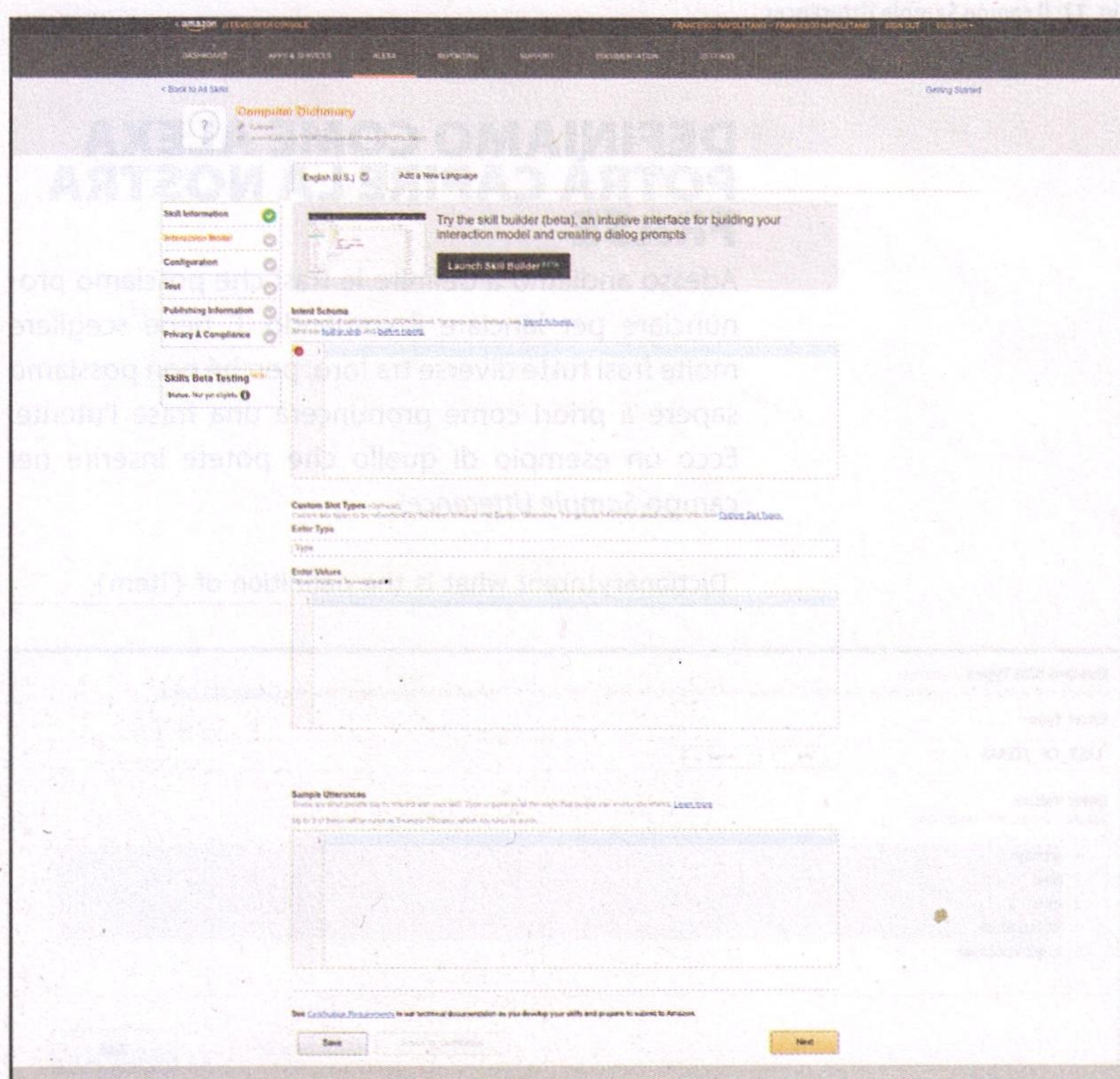
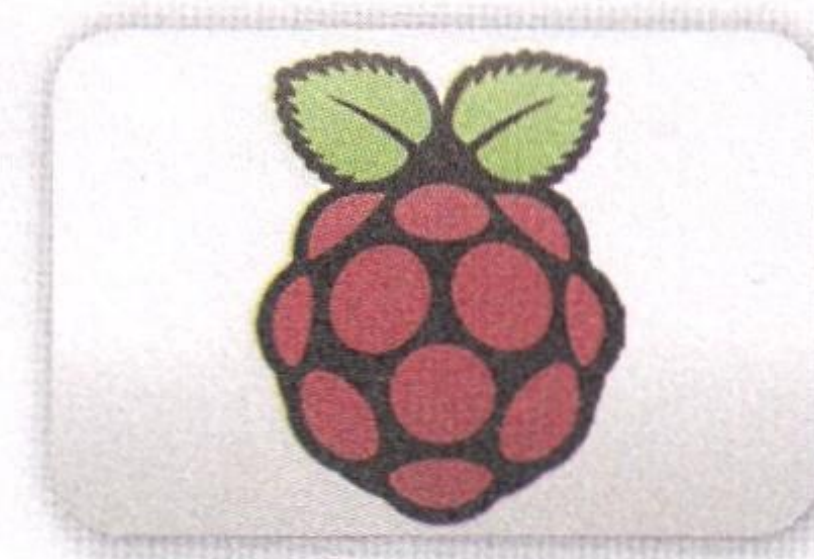



Fig. 11: Il modulo per la definizione degli schemi della nostra skill

AMAZON PAGA GLI SVILUPPATORI DI SKILL

La fame di "skill" per Alexa è così diffusa che Amazon paga gli sviluppatori delle migliori skill per Alexa. Un'occasione da non lasciarsi sfuggire. https://is.gd/amazonpaga

Cliccate *Save* e poi *Next*. La nostra skill sarà lanciata con il comando *Alexa, ask computer terms…* che come potete vedere è proprio il parametro *Invocation Name* che abbiamo definito. Ci troviamo ora davanti ad un nuovo form. Esaminiamolo insieme:

Copiate questo JSON all'interno del primo campo, *Intent Schema*.

```
{"intents": [
    {"intent": "DictionaryIntent", "slots": [{"name" :
                    "Item", "type": "LIST_OF_ITEMS"}]},
    {"intent": "AMAZON.RepeatIntent"},
    {"intent": "AMAZON.HelpIntent"},
    {"intent": "AMAZON.StopIntent"},
    {"intent": "AMAZON.CancelIntent"}
  ]
}
```

Che cos'è? È quella che potremmo definire la "lista dei comandi" della nostra skill. La prima riga contiene un riferimento a *DictionaryIntent*, con un parametro Item di tipo *LIST_OF_ITEMS*. Questo sarà il parametro che passeremo ad Alexa e sarà contenuto in una lista che definiremo a brevissimo.

Gli altri 4 Intent sono relativi a comandi di Stop, Cancella, Aiuto e Ripeti, come potete facilmente intuire.

NOTA

APPLE HOMEPOD

Ovviamente Apple non poteva stare a guardare, e ha da poco presentato Apple HomePod, che potete trovare a questo indirizzo. https://www.apple.com/homepod/ Purtroppo, come gli altri prodotti visti nel corso dell'articolo, non è ancora disponibile in Italia.

Sample Utterances
These are what people say to interact with your skill. Type or paste in all the ways that people can invoke the intents. Learn more
Up to 3 of these will be used as Example Phrases, which are hints to users.

```
1 RecipeIntent what is the definition of {Item}
2 RecipeIntent what does {Item} means
3 RecipeIntent what {Item} means
```

Fig. 12: Il campo Sample Utterances

DEFINIAMO COME ALEXA POTRÀ CAPIRE LA NOSTRA FRASE

Adesso andiamo a definire le frasi che possiamo pronunciare per lanciare il comando. È bene scegliere molte frasi tutte diverse tra loro, perché non possiamo sapere a priori come pronuncerà una frase l'utente. Ecco un esempio di quello che potete inserire nel campo *Sample Utterances*.

```
DictionaryIntent what is the definition of {Item}
DictionaryIntent what does {Item} mean
DictionaryIntent what {Item} means
```

Fig. 13: I campi "Custom Slot Type"

Scatenate la fantasia e aggiungetene quante più vi aggrada, ricordandovi di cominciare sempre con *DictionaryIntent* e di includere al posto giusto *{Item}*.

INSERIAMO LA LISTA DI POSSIBILI PAROLE DA DEFINIRE

Abbiamo impostato come Alexa potrà capire la nostra domanda, ora dobbiamo dire ad Alexa quali parole aspettarsi. Compiliamo quindi il famoso *LIST_OF_ITEMS* che abbiamo già visto in precedenza. Compilate i due campi in *Custom Slot Type* come in figura. Ovviamente, più termini inserirete, più definizioni dovrete dare, per ora rimaniamo con 5 termini:

- *array*
- *mvc*
- *cpu*
- *recursion*
- *ioprogrammo*

Cliccate *Add* e poi *Save*. Il nostro *Interaction Model* verrà creato (ci potrebbe volere qualche minuto).

CREIAMO LA LOGICA PER LA NOSTRA SKILL SU AWS LAMBDA

Mettiamo per un attimo da parte la configurazione della nostra skill e passiamo ad AWS Lambda.

Fig. 14: Homepage di Amazon AWS

Andate su AWS all'indirizzo https://aws.amazon.com e registratevi se ancora non l'avete fatto. L'account e l'uso dei servizi è gratuito per gli utenti con soglie abbastanza alte, come potete vedere qui: https://aws.amazon.com/it/free/ ma potrebbe essere richiesto di inserire una carta di credito a garanzia.

Nella lista infinita di servizi disponibili cercate *Lambda* e createne una. Attenzione, come regione per le vostre Lambda, in alto a destra, selezionate *US EAST* (se avete creato una skill usando *EN-us* come lingua) o *IRELAND* (se avete usato *en-gb*).

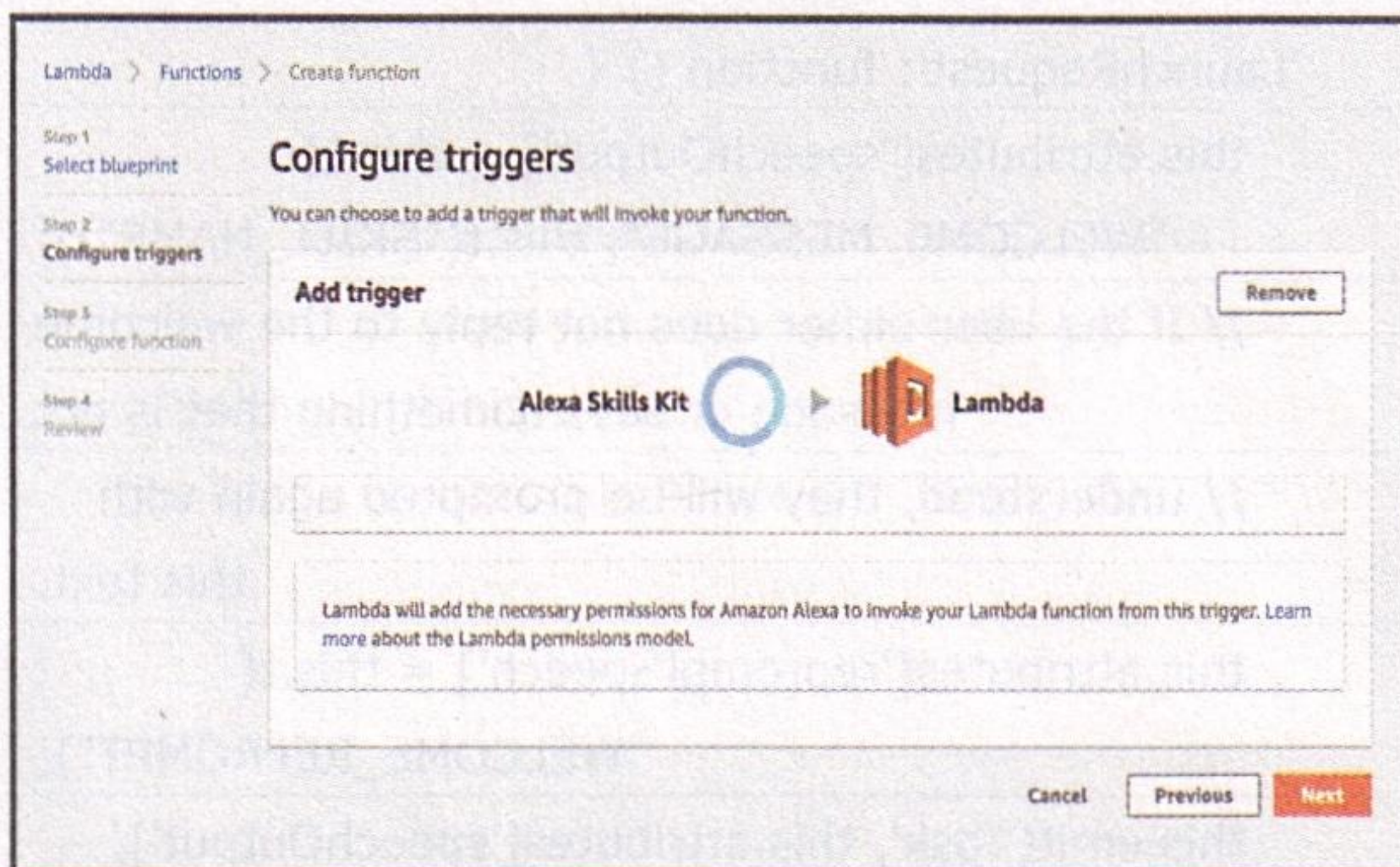



Fig. 15: Creazione della nostra Lambda

Cliccate *Author from Scratch* e nella schermata *Add Trigger* selezionate *Alexa Skill Kit*. Se non la trovate nella lista di possibili trigger, vuol dire che la regione selezionata non è una delle due che abbiamo spiegato poc'anzi. Cliccate *Next*. Configureremo adesso il codice della Lambda e un ruolo di esecuzione.

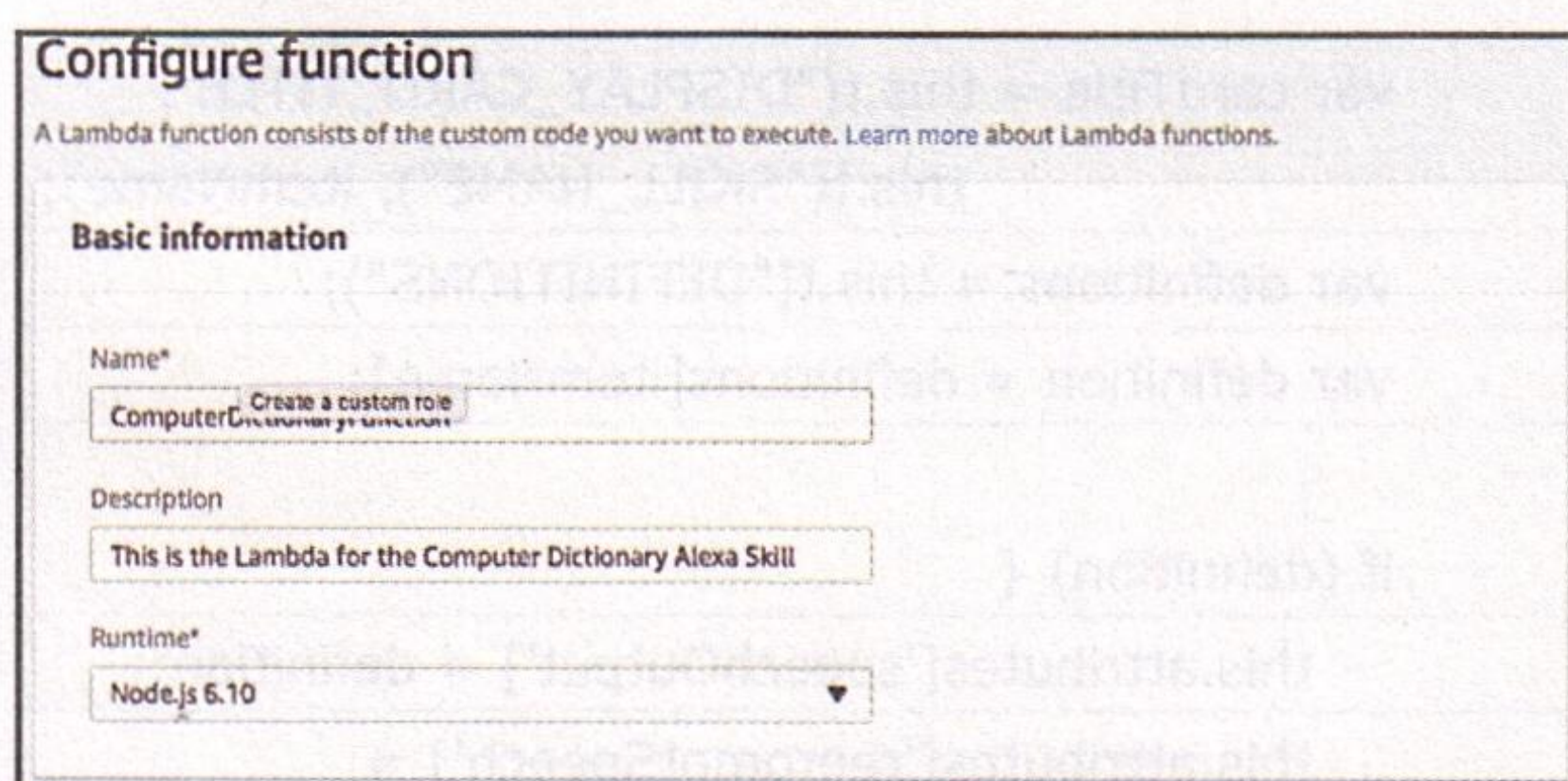



Fig. 16: Informazioni di base

Compilate la prima parte del modulo con un nome per la vostra funzione e una descrizione. Come *Runtime* scegliete *Node.js 6.10*. Scrollate in basso nella pagina: lasciate l'handler così com'è e nella tendina *Role* scegliete *Custom Role* per creare un nuovo ruolo che avrà i permessi di esecuzione per la Lambda. Nella nuova

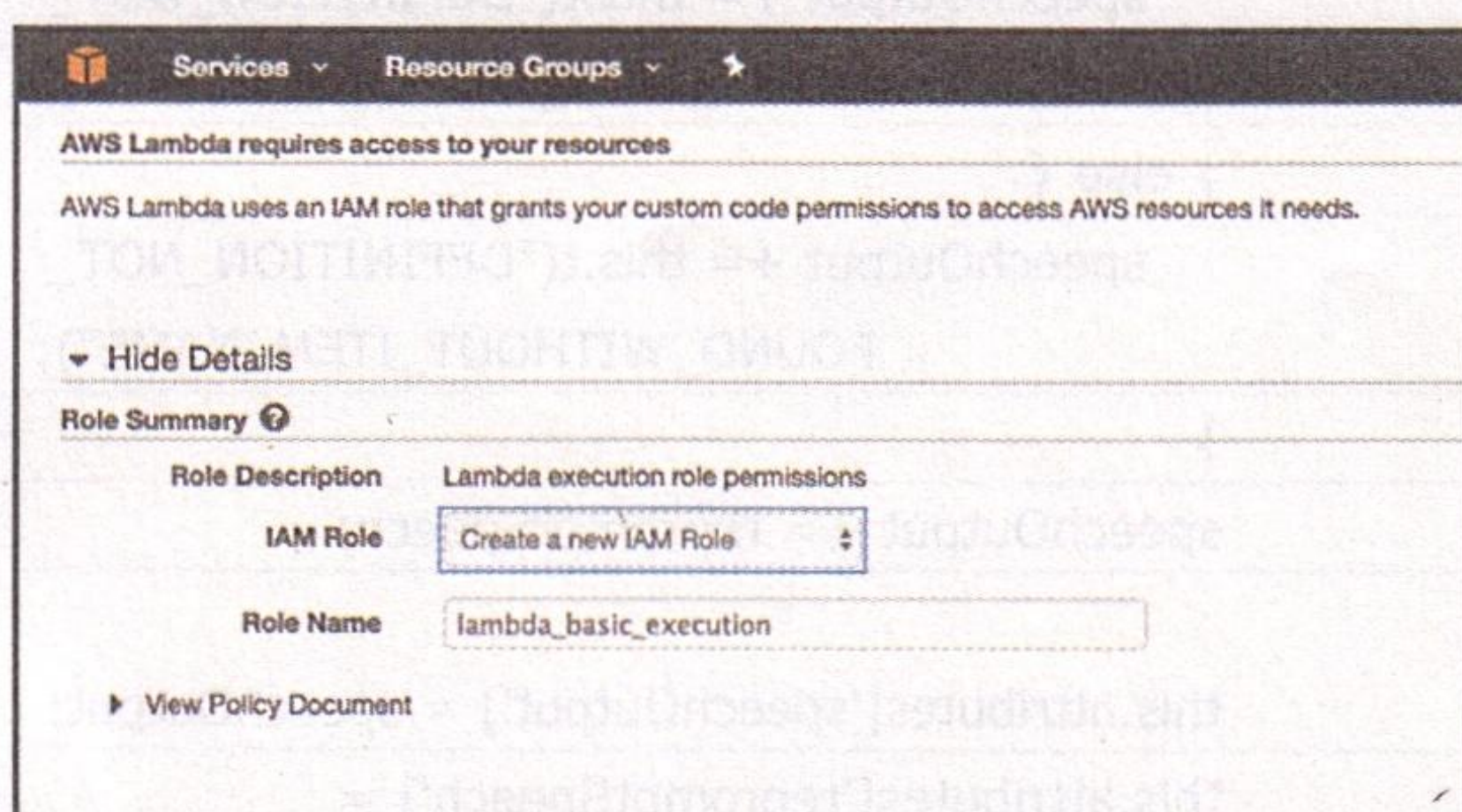
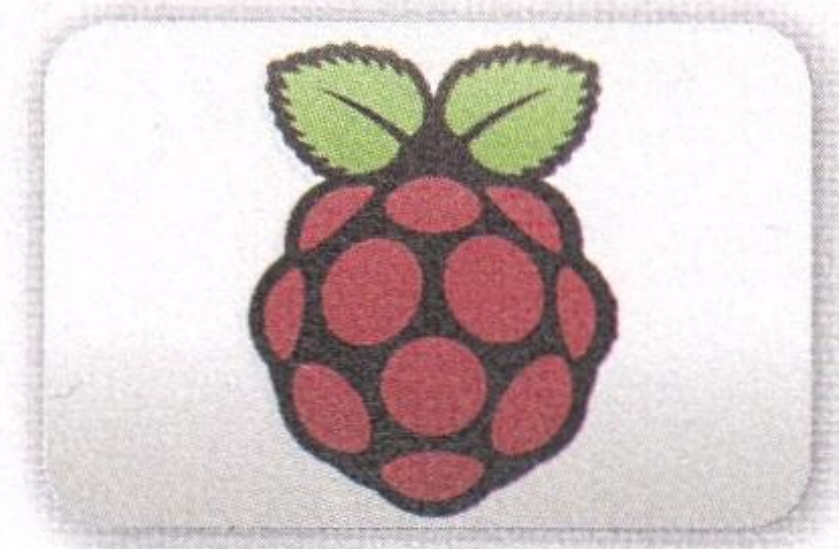



Fig. 17: Ruolo di esecuzione

finestra che si aprirà scegliete un nome e cliccate *Allow*. La finestra si chiuderà e tornerete alla pagina di definizione della Lambda. Lasciate gli *advanced settings* così come sono e cliccate *Next*. Cliccate poi *Create Function*. La vostra Lambda è pronta, anche se non fa nulla.

Copiate l'ARN della Lambda, in alto a destra e tornate nella configurazione della Skill su https://developer.amazon.com.

AGGIUNGIAMO LA LAMBDA ALLA NOSTRA SKILL

Modifichiamo la nostra Skill e aggiungiamo la Lambda tra i *Global Fields*. Mi raccomando, se avete scelto *EN-US* come lingua, la lambda dovrà essere stata creata nella regione *US – EAST* (stessa cosa per *EN-GB*, la lambda dovrà essere stata creata nella regione *IRELAND*). Torniamo ora al codice vero e proprio. Lo caricheremo su AWS una volta completato. Ci basti sapere che la lambda è correttamente settata in developer.amazon.com.

IL CODICE DELLA NOSTRA LAMBDA

Come abbiamo visto, abbiamo settato degli Intent per la nostra skill, è giunto ora il momento di metterli in pratica, scrivendo il codice della nostra lamba. Cominciamo col creare in una cartella i file *package.json, index.js* e *definitions.js*.

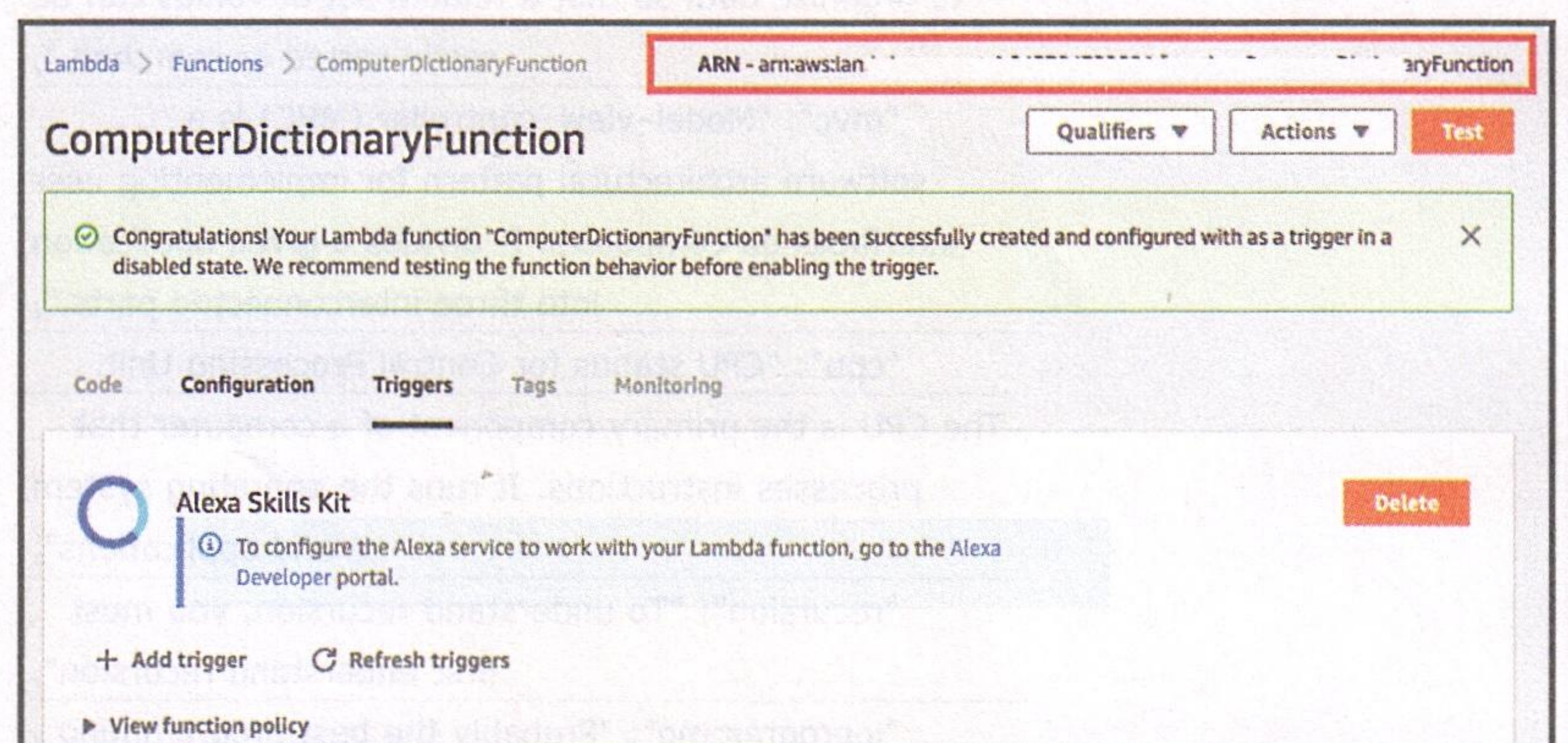



Fig. 18: ARN

package.json è il file di setup della nostra applicazione ed è così formato:

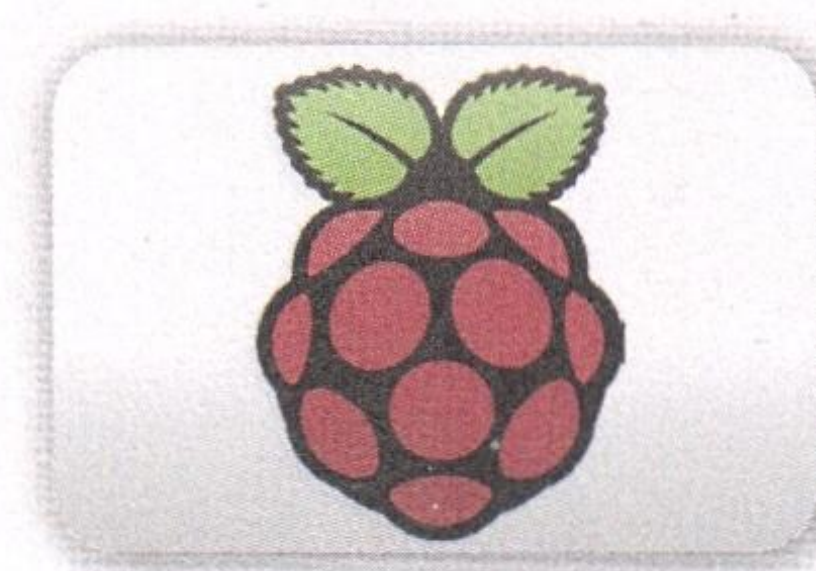
```
{
  "name": "computer dictionary",
  "description": "Computer Dictionary skill",
  "version": "0.0.1",
  "dependencies": {
    "alexa-sdk": "^1.0.6"
  }
}
```

Include, ovviamente, l'SDK di Alexa e poco altro. *definitions.js* includerà le definizioni del nostro dizionario, divise per lingua:

```
module.exports = {
    "DEFINITION_EN_GB" : {
        "array": "An array is a data structure that
    contains a group of elements. Typically these elements
        are all of the same data type, such as an integer or
string. Arrays are commonly used in computer programs
```

```
to organize data so that a related set of values can be
easily sorted or searched.",
    "mvc": "Model-view-controller (MVC) is a
software architectural pattern for implementing user
interfaces on computers. It divides a given application
into three interconnected parts.",
    "cpu": "CPU stands for Central Processing Unit.
The CPU is the primary component of a computer that
processes instructions. It runs the operating system and
applications",
    "recursion": "To understand recursion, you must
first understand recursion",
    "ioprogrammo": "Probably the best programming
magazine you can find in Italy"
  },
  "DEFINITION_EN_US" : {
    "array": "An array is a data structure that
contains a group of elements. Typically these elements
are all of the same data type, such as an integer or
string. Arrays are commonly used in computer programs
to organize data so that a related set of values can be
easily sorted or searched.",
    "mvc": "Model-view-controller (MVC) is a
software architectural pattern for implementing user
interfaces on computers. It divides a given application
into three interconnected parts.",
    "cpu": "CPU stands for Central Processing Unit.
The CPU is the primary component of a computer that
processes instructions. It runs the operating system
and applications",
    "recursion": "To understand recursion, you must
first understand recursion",
    "ioprogrammo": "Probably the best programming
magazine you can find in Italy"
  }
};
```

Il file *index.js* include l'applicazione vera e propria. Vediamolo nel suo insieme, lo analizzeremo pezzo per pezzo.

```
'use strict';

var Alexa = require('alexa-sdk');
var APP_ID = 'ID_DELLA_TUA_APPLICAZIONE';
var definitions = require('./definitions');

exports.handler = function(event, context, callback) {
  var alexa = Alexa.handler(event, context);
  alexa.APP_ID = APP_ID;
  // To enable string internationalization (i18n)
features, set a resources object.
  alexa.resources = languageStrings;
  alexa.registerHandlers(handlers);
  alexa.execute();
};

var handlers = {
  'LaunchRequest': function () {
    this.attributes['speechOutput'] = this.t(
      "WELCOME_MESSAGE", this.t("SKILL_NAME"));
    // If the user either does not reply to the welcome
message or says something that is not
    // understood, they will be prompted again with
this text.
    this.attributes['repromptSpeech'] = this.t(
"WELCOME_REPROMPT");
    this.emit(':ask', this.attributes['speechOutput'],
this.attributes['repromptSpeech'])
  },
  'DictionaryIntent': function () {
    var itemSlot = this.event.request.intent.slots.Item;
    var itemName;
    if (itemSlot && itemSlot.value) {
      itemName = itemSlot.value.toLowerCase();
    }

    var cardTitle = this.t("DISPLAY_CARD_TITLE",
this.t("SKILL_NAME"), itemName);
    var definitions = this.t("DEFINITIONS");
    var definition = definitions[itemName];

    if (definition) {
      this.attributes['speechOutput'] = definition;
      this.attributes['repromptSpeech'] =
        this.t("DEFINITION_REPEAT_MESSAGE");
      this.emit(':tellWithCard', definition, this.
attributes['repromptSpeech'], cardTitle, definition);
    } else {
      var speechOutput = this.t("DEFINITION_NOT_
FOUND_MESSAGE");
      var repromptSpeech = this.t("DEFINITION_
NOT_FOUND_REPROMPT");
      if (itemName) {
        speechOutput += this.t("DEFINITION_NOT_
FOUND_WITH_ITEM_NAME", itemName);
      } else {
        speechOutput += this.t("DEFINITION_NOT_
FOUND_WITHOUT_ITEM_NAME");
      }
      speechOutput += repromptSpeech;

      this.attributes['speechOutput'] = speechOutput;
      this.attributes['repromptSpeech'] =
repromptSpeech;

      this.emit(':ask', speechOutput,
repromptSpeech);
    }
  },
  'AMAZON.HelpIntent': function () {
    this.attributes['speechOutput'] = this.t(
"HELP_MESSAGE");
    this.attributes['repromptSpeech'] = this.t(
"HELP_REPROMPT");
    this.emit(':ask', this.attributes['speechOutput'],
```

```
                        this.attributes['repromptSpeech'])
  },
  'AMAZON.RepeatIntent': function ()
  {
    this.emit(':ask', this.attributes['speechOutput'],
                        this.attributes['repromptSpeech'])
  },
  'AMAZON.StopIntent': function ()
  {
    this.emit('SessionEndedRequest');
  },
  'AMAZON.CancelIntent': function ()
  {
    this.emit('SessionEndedRequest');
  },
  'SessionEndedRequest':function ()
  {
    this.emit(':tell', this.t("STOP_MESSAGE"));
  },
  'Unhandled': function ()
  {
    this.attributes['speechOutput'] = this.t("HELP_
                                          MESSAGE");
    this.attributes['repromptSpeech'] = this.t("HELP_
                                          REPROMPT");
    this.emit(':ask', this.attributes['speechOutput'],
                        this.attributes['repromptSpeech'])
  }
};

var languageStrings = {
  "en": {
    "translation": {
      "DEFINITIONS": definitions.DEFINITION_EN_US,
      "SKILL_NAME": "Computer Dictionary",
      "WELCOME_MESSAGE": "Welcome to %s. You
  can ask a question like, what\'s the meaning of array?
                        ... Now, what can I help you with.",
      "WELCOME_REPROMPT": "For instructions on
                  what you can say, please say help me.",
      "DISPLAY_CARD_TITLE": "%s  - Recipe for %s.",
      "HELP_MESSAGE": "You can ask questions
  such as, what\'s the meaning of array, or, you can say
                  exit...Now, what can I help you with?",
      "HELP_REPROMPT": "You can say things like,
what\'s the meaning of array, or you can say exit...Now,
                        what can I help you with?",
      "STOP_MESSAGE": "Goodbye!",
      "DEFINITION_REPEAT_MESSAGE": "Try saying
                                          repeat.",
      "DEFINITION_NOT_FOUND_MESSAGE": "I\'m
                  sorry, I currently do not know",
      "DEFINITION_NOT_FOUND_WITH_ITEM_
                  NAME": "the definition for %s. ",
      "DEFINITION_NOT_FOUND_WITHOUT_ITEM_
                  NAME": "that definition. ",
      "DEFINITION_NOT_FOUND_REPROMPT": "What
                  else can I help with?"
    }
  },
  "en-US":
  {
    "translation":
    {
      "DEFINITIONS" : definitions.DEFINITION_EN_US,
      "SKILL_NAME": "Computer Dictionary"
    }
  },
  "en-GB":
  {
    "translation":
    {
      "DEFINITIONS": definitions.DEFINITION_EN_GB,
      "SKILL_NAME": "Computer Dictionary"
    }
  }
};
```

Il primo pezzo di codice non è nient'altro che il setup della nostra applicazione:

```
'use strict';

var Alexa = require('alexa-sdk');
var APP_ID = 'ID_DELLA_TUA_APPLICAZIONE';
var definitions = require('./definitions');

exports.handler = function(event, context, callback) {
  var alexa = Alexa.handler(event, context);
  alexa.APP_ID = APP_ID;
  // To enable string internationalization (i18n)
                  features, set a resources object.
  alexa.resources = languageStrings;
  alexa.registerHandlers(handlers);
  alexa.execute();
};
```

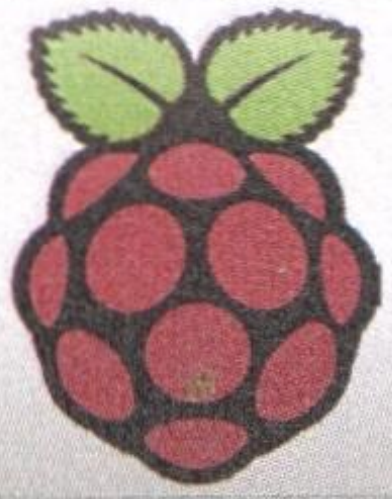
Includiamo il file delle definizioni, una serie di stringhe che contengono i messaggi di errore ed esportiamo l'handler che sarà eseguito dalla lambda. Registriamo i diversi handler che eseguiranno la logica della nostra lambda e che abbiamo definito poco sotto nell'oggetto handlers in alexa e lanciamo il metodo *execute()*.

L'OGGETTO HANDLERS

Vi ricordate la prima definizione della nostra skill? Era un file JSON contenenti una serie di Intent. Questi Intent sono ora in handlers.

```
  'AMAZON.HelpIntent': function () {
    this.attributes['speechOutput'] = this.t("HELP_
                                          MESSAGE");
    this.attributes['repromptSpeech'] = this.t(
                                    "HELP_REPROMPT");
```

ALEXA SDK

Potete trovare tutto il codice sviluppato da Amazon per Alexa all'indirizzo

https://github.com/alexa/

```
        this.emit(':ask', this.attributes['speechOutput'],
                        this.attributes['repromptSpeech'])
    },
    'AMAZON.RepeatIntent': function () {
        this.emit(':ask', this.attributes['speechOutput'],
                        this.attributes['repromptSpeech'])
    },
    'AMAZON.StopIntent': function () {
        this.emit('SessionEndedRequest');
    },
    'AMAZON.CancelIntent': function () {
        this.emit('SessionEndedRequest');
    },
```

Questi sono handler standard forniti da Amazon, ma quello che ci interessa di più è il *DictionaryIntent*. Riceverà come parametro il nostro Item e lo userà come chiave per cercare nelle definizioni la corretta definizione del termine. Se trovata, Alexa pronuncerà la definizione, altrimenti chiederà di ripetere o comunicherà che la definizione non è stata trovata.

```
'DictionaryIntent': function () {
    var itemSlot = this.event.request.intent.slots.
                                            Item;
    var itemName;
    if (itemSlot && itemSlot.value) {
        itemName = itemSlot.value.toLowerCase();
    }

    var cardTitle = this.t("DISPLAY_CARD_TITLE",
                    this.t("SKILL_NAME"), itemName);
    var definitions = this.t("DEFINITIONS");
    var definition = definitions[itemName];

    if (definition) {
        this.attributes['speechOutput'] = definition;
        this.attributes['repromptSpeech'] =
                this.t("DEFINITION_REPEAT_MESSAGE");
        this.emit(':tellWithCard', definition, this.
    attributes['repromptSpeech'], cardTitle, definition);
    } else {
        var speechOutput = this.t("DEFINITION_NOT_
                                    FOUND_MESSAGE");
        var repromptSpeech = this.t("DEFINITION_
                                NOT_FOUND_REPROMPT");
        if (itemName) {
            speechOutput += this.t("DEFINITION_NOT_
            FOUND_WITH_ITEM_NAME", itemName);
        } else {
            speechOutput += this.t("DEFINITION_NOT_
                        FOUND_WITHOUT_ITEM_NAME");
        }
        speechOutput += repromptSpeech;

        this.attributes['speechOutput'] = speechOutput;
        this.attributes['repromptSpeech'] =
                                    repromptSpeech;

        this.emit(':ask', speechOutput, repromptSpeech);
    }
},
```

COPIARE IL NOSTRO CODICE SU AWS

Una volta completato il codice, dobbiamo copiarlo su AWS. Il sito permette l'upload di file, quindi basterà compattare il codice in un file .zip e caricarlo. Andate nella cartella dove avete creato i file. Lanciate il comando *npm install*. Alla fine dell'esecuzione, aprite la cartella *node_modules* e copiate tutto quel che vi trovate nella root del vostro progetto. Potete anche cancellare *node_modules*. Compattate il tutto in un bel file zip e caricatelo su AWS, ricordandovi di cliccare il pulsante *Save* per cominciare l'upload.

TESTARE LA NOSTRA APPLICAZIONE

All'interno del pannello di gestione delle skills, potete testare il vostro comando e vedere cosa viene "scambiato" tra alexa e la lambda. Cliccate *Ask Computer Dictionary* dopo aver inserito una domanda.

Ecco un esempio di richiesta: notate la chiamata a

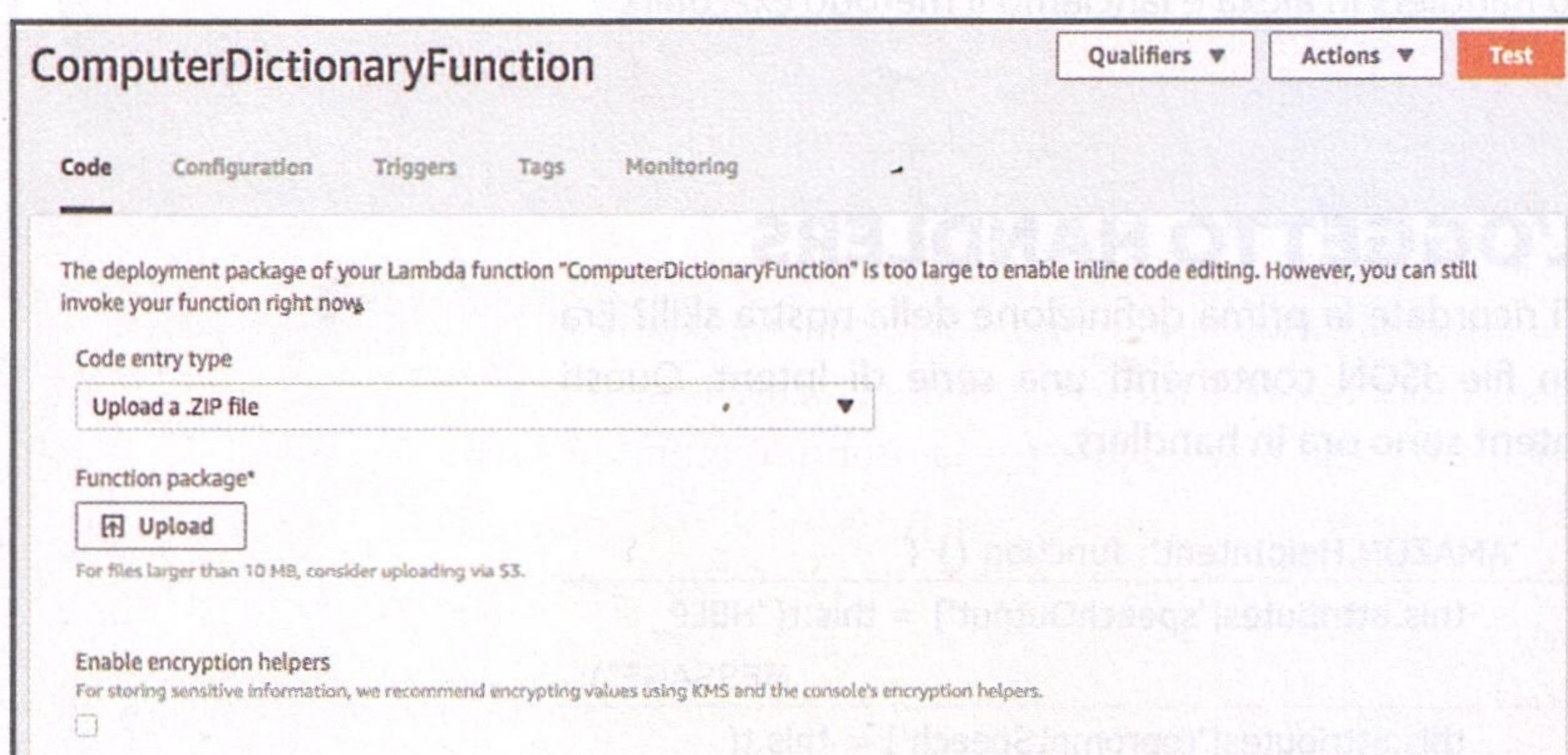



Fig. 19: Upload AWS

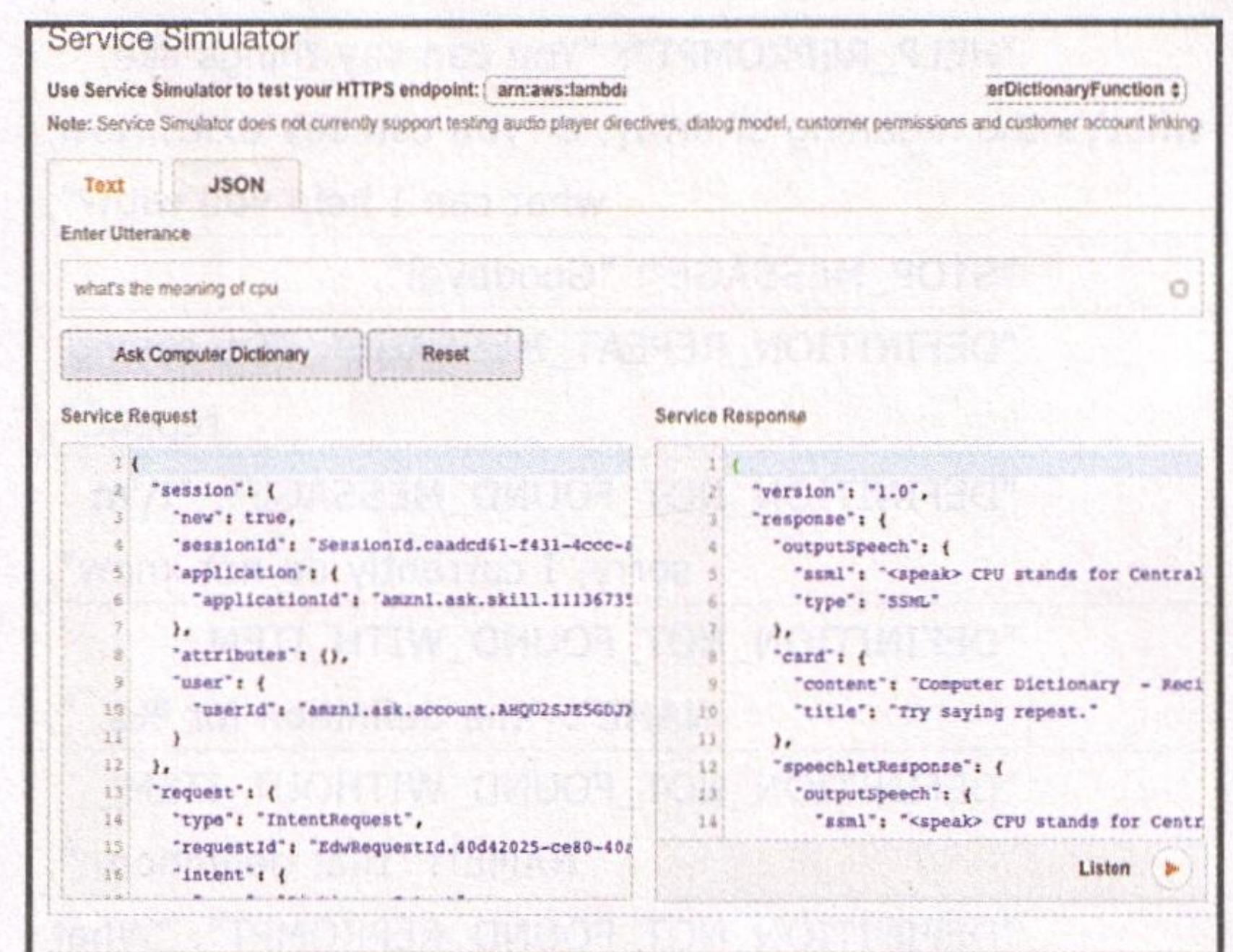



Fig. 20: il nostro test

DictionaryIntent e l'Item passato, che sarà CPU.

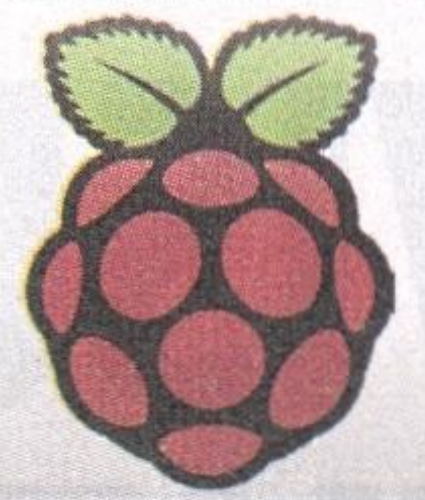

```
{
  "session": {
    "new": true,
    "sessionId": "SessionId.caadcd61",
    "application": {
      "applicationId": "amzn1.ask.skill.11136735-cce4-
                              4974-8a1b-1f1973c39be3"
    },
    "attributes": {},
    "user": {
      "userId": "amzn1.ask.account.ALEEBCO3NLOA"
    }
  },
  "request": {
    "type": "IntentRequest",
    "requestId": "EdwRequestId.40d42025-ce80-40ab-
                              9827-b455f6d180d9",
    "intent": {
      "name": "DictionaryIntent",
      "slots": {
        "Item": {
          "name": "Item",
          "value": "CPU"
        }
      }
    },
    "locale": "en-GB",
    "timestamp": "2017-08-26T16:34:03Z"
  },
  "context": {
    "AudioPlayer": {
      "playerActivity": "IDLE"
    },
    "System": {
      "application": {
        "applicationId": "amzn1.ask.skill.11136735-cce4-
                              4974-8a1b-1f1973c39be3"
      },
      "user": {
        "userId": "amzn1.ask.account.AHQU2SJE3NLOA"
      },
      "device": {
        "supportedInterfaces": {}
      }
    }
  },
  "version": "1.0"
}
```


La risposta, conterrà la definizione che verrà letta da Alexa.


```
{
  "version": "1.0",
  "response": {
    "outputSpeech": {
      "ssml": "<speak> CPU stands for Central
Processing Unit. The CPU is the primary component of a
computer that processes instructions. It runs the
operating system and applications </speak>",
      "type": "SSML"
    },
    "card":
    {
      "content": "Computer Dictionary  - Recipe for cpu.",
      "title": "Try saying repeat."
    },
    "speechletResponse":
    {
      "outputSpeech":
      {
        "ssml": "<speak> CPU stands for Central
Processing Unit. The CPU is the primary component of a
computer that processes instructions. It runs the opera
ting system and applications </speak>"
      },
      "card": {
        "content": "Computer Dictionary  - Recipe for cpu.",
        "title": "Try saying repeat."
      },
      "shouldEndSession": true
    }
  },
  "sessionAttributes": {
    "speechOutput": "CPU stands for Central Processing
Unit. The CPU is the primary component of a computer
that processes instructions. It runs the operating system
and applications",
    "repromptSpeech": "Try saying repeat."
  }
}
```


Avete un device Echo registrato con la stessa email con cui avete creato la vostra skill? Allora potete già testare dal vivo la vostra skill! Andate vicino al vostro Amazon Echo e chiedete "Alexa, ask computer dictionary what's the meaning of CPU" e otterrete una risposta!

CONCLUSIONI

Gli assistenti vocali, come abbiamo visto sono rivoluzionari. Possiamo usarli per comandare luci, termostati e altri componenti di casa, possiamo giocarci, possiamo usarli per ascoltare e condividere contenuti o ottenere informazioni, il tutto solamente grazie alla nostra voce. Google, Amazon, Apple e Samsung si stanno sfidando in questo mercato appena nato, ma già ricco di attrattiva, per gli sviluppatori che salteranno sul treno per tempo (in Italia Alexa sta per arrivare), ci saranno ottime prospettive.

Francesco Napoletano

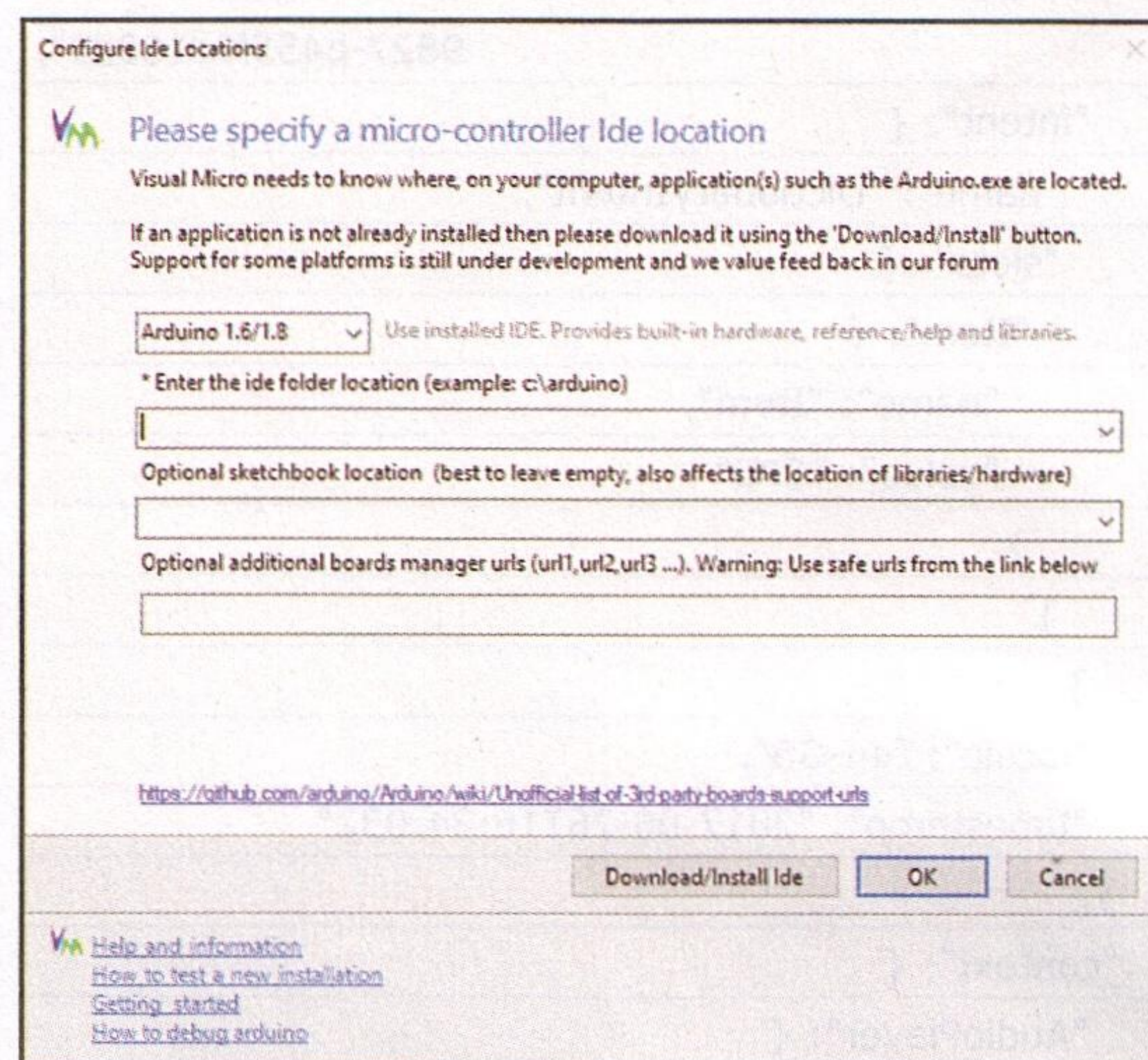
Francesco Napoletano, "napolux" sul Web, 37 anni, lavora come software engineer (web e mobile) per una big della Silicon Valley. Quando non lavora, gioca ai videogame, ascolta musica degli anni '80 e viaggia.
https://napolux.com

VISUAL STUDIO E ARDUINO: CHE COPPIA!

IMPARIAMO A CREARE E GESTIRE IL CODICE DEGLI SKETCH DI ARDUINO DIRETTAMENTE NELL'IDE DI VISUAL STUDIO 2017. REALIZZIAMO INOLTRE UN PROGETTO IN VISUAL BASIC PER INTERAGIRE CON QUESTA SCHEDA

Abbiamo già visto nelle pagine di ioProgrammo alcuni articoli e progetti sviluppati con Arduino, un microcontroller che costituisce la base di una piattaforma hardware modulare in grado di sprigionare tutta la nostra fantasia nella creazione di dispositivi dotati di sensori e altri componenti elettronici. Ovviamente abbiamo visto anche innumerevoli articoli dedicati a Visual Studio e ai suoi linguaggi. Mai prima d'ora, però, abbiamo messo insieme i due mondi Arduino e Visual Studio: un insieme a dir poco dirompente!

ARDUINO IDE FOR VISUAL STUDIO

Per poter lavorare con Arduino in Visual Studio dovete fare due operazioni. La prima è verificare che nella vostra installazione di Visual Studio sia stato incluso il modulo relativo al supporto C++. Se non c'è, bisogna avviare Visual Studio Installer dal menu Start di Windows 10 e aggiungerlo. La seconda operazione è procurarvi l'add-in Arduino IDE for Visual Studio. Questo lo potete trovare direttamente al link http://bit.ly/getArduinoIDE oppure passando dal menu di Visual Studio 2017: *Tools > Extensions and Updates > sezione Online*. Nella casella di ricerca inserite la stringa "Arduino IDE" (compresi i doppi apici) e pochi istanti dopo apparirà l'add-in (**fig. 1**). Non appena avrete completato il download, nella zona

inferiore della finestra apparirà un messaggio per informarvi che dovete chiudere tutte le istanze di Visual Studio per poter procedere con l'installazione. Pochi istanti dopo che avrete chiuso Visual Studio partirà l'installazione con VSIX Installer: cliccate sul pulsante *Modifica* e

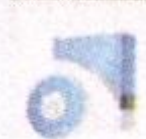
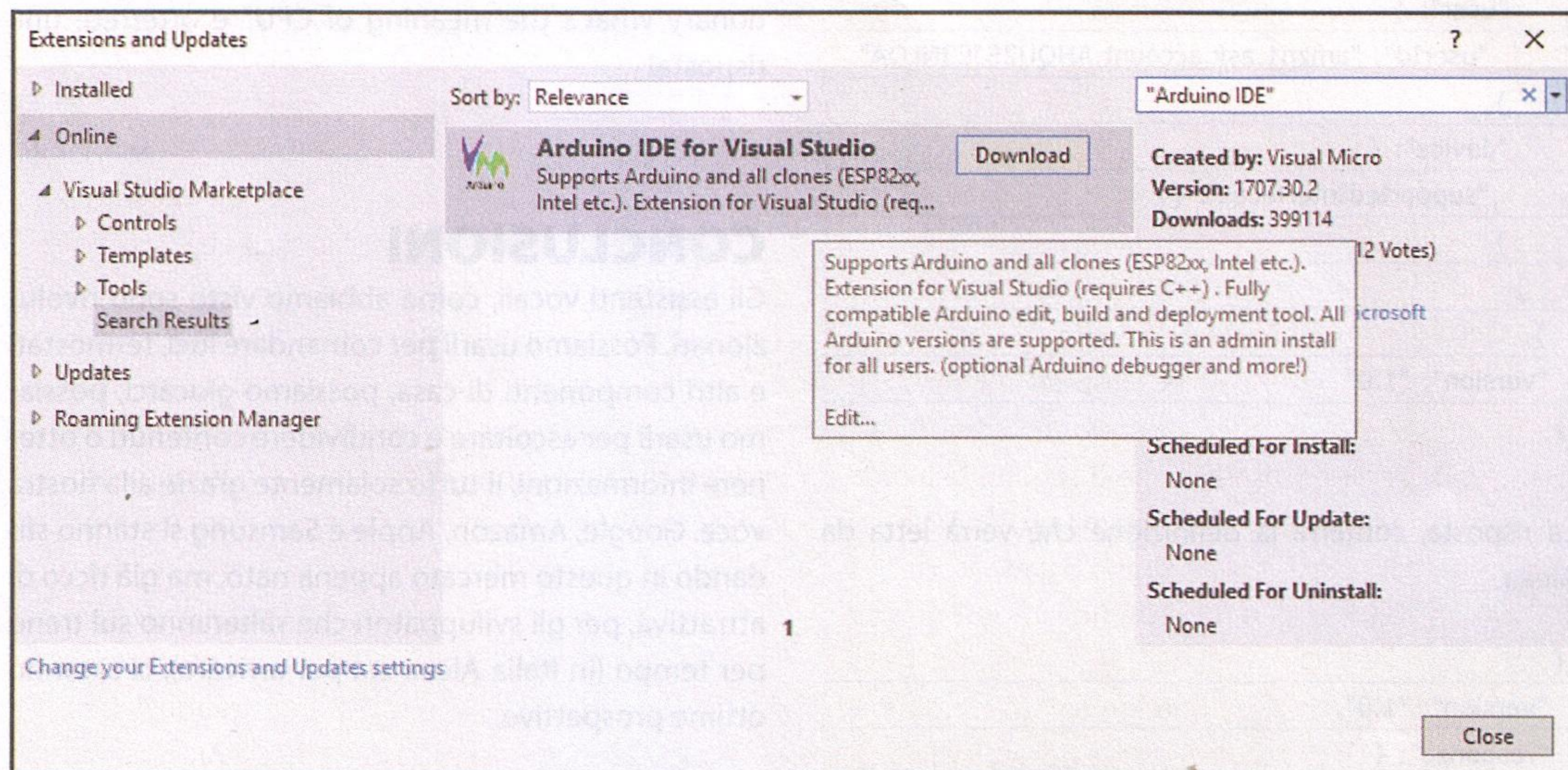



Fig. 1: Selezione dell'add-in "Arduino IDE for Visual Studio"

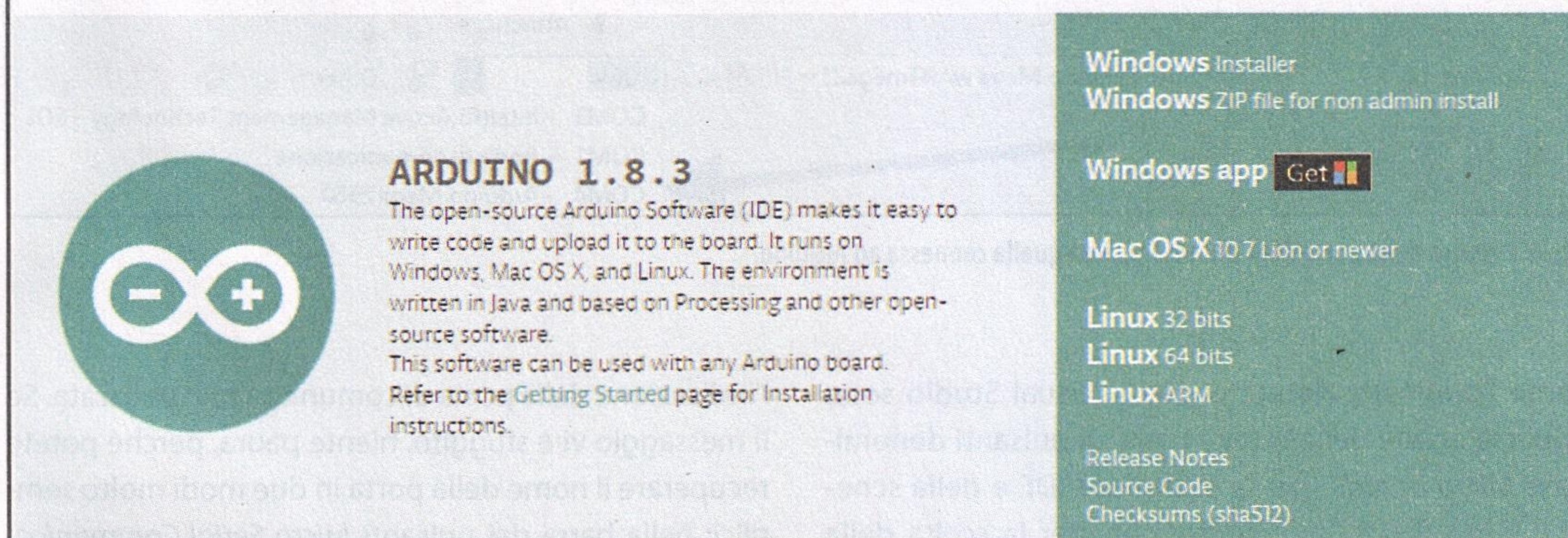



Fig. 3: La classica finestra di Arduino IDE con lo sketch di base.

il gioco è fatto. Una volta installato l'add-in, avviate Visual Studio: apparirà una finestra come quella in **fig. 2**. La prima casella a discesa è configurata sull'ultima versione di Arduino IDE (1.6/1.8) e pertanto potete lasciarla così com'è. La terza e la quarta casella sono opzionali e non è necessario indicare nulla, salvo che non abbiate particolari esigenze che vanno oltre l'uso ordinario. La seconda casella invece è obbligatoria e, come indicato nelle note riportate in alto nella finestra, deve contenere il percorso dell'eseguibile dell'IDE di Arduino. Per quale motivo? È semplice: l'add-in per Visual Studio funziona come "ponte" verso il vero IDE di Arduino e quindi rende necessario installare anche l'IDE comunemente utilizzato. Lasciamo momentaneamente in sospeso la configurazione dell'add-in di Visual Studio e avviamo l'installazione dell'IDE di Arduino, cliccando sul pulsante *Download/Install Ide*. Questa operazione aprirà il vostro browser Internet alla pagina https://www.arduino.cc/en/Main/Software#, dove potrete trovare tutti i pacchetti di installazione disponibili: per Windows 10 (attraverso Microsoft Store), per altre versioni di Windows, per Mac OS X e per Linux. Dato che la versione scaricata da Microsoft Store si installa in una cartella nascosta e protetta che tipicamente si trova sotto *C:\Program Files\WindowsApps*, ottenere il nome completo della cartella e poterlo quindi utilizzare diventa veramente arduo se non impossibile. Vi consigliamo quindi di scaricare la versione tradizionale per Windows (Zip o Installer): dalla procedura di installazione prendete nota del percorso dell'applicazione Arduino perché ci servirà dopo (nel nostro caso è stato *C:\Program Files (x86)\Arduino*, ma se volete potete modificare la cartella di destinazione, per esempio impostandola a *C:\Arduino*). Avviate l'installazione e seguite le indicazioni finché non otterrete la conferma finale. A questo punto avviate il programma Arduino dal menu Start di Windows 10 e otterrete la classica finestra come nella **fig. 3**.

Ora dobbiamo terminare la configurazione dell'add-in Arduino IDE for Visual Studio: inserite nella seconda casella il percorso di Arduino (per esempio: *C:\Program Files (x86)\Arduino\Arduino.exe*) e cliccate sul pulsante *OK*. Si aprirà la finestra di Output con alcune indicazioni che non ci interessano particolarmente e apparirà anche un nuovo menu di nome vMicro (Fig. 4).

Con questo nuovo menu possiamo fare tutto quello

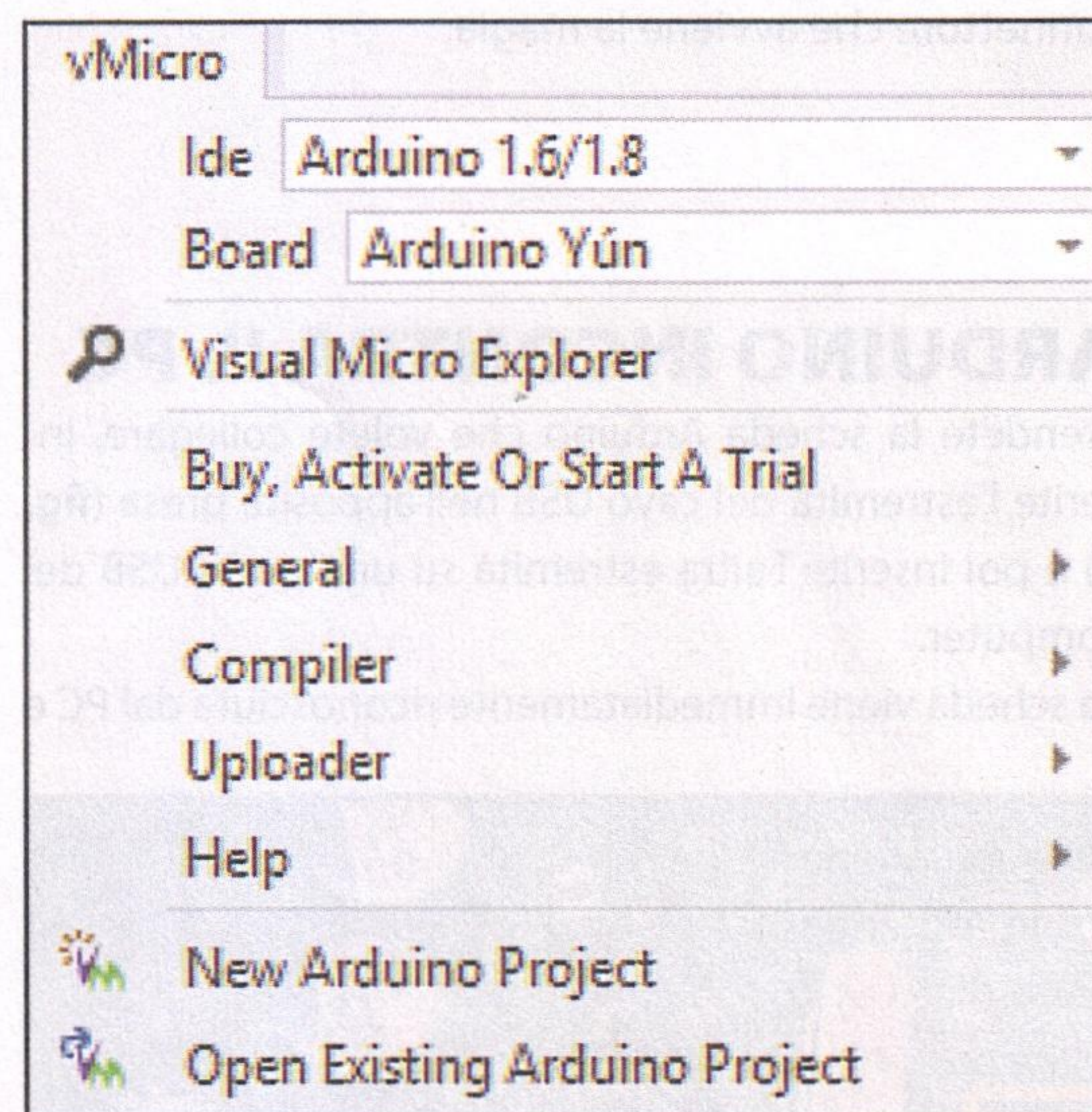



Fig. 4: Il nuovo menu vMicro di Visual Studio 2017.

che potremmo fare con il normale IDE di Arduino: configurare la versione di Arduino, caricare gli sketch su Arduino, aprire gli esempi già pronti di sketch e così via. La prima cosa che dobbiamo fare con questo menu è appunto configurare la nostra scheda di Arduino, visto che ne esistono molte versioni e bisogna che l'IDE lavori con la nostra scheda specifica. A questo proposito vedrete che tutti i nomi di scheda si riferiscono esclusivamente ad Arduino ma non ai molti cloni esistenti. A questo proposito non dovete preoccuparvi, perché tutti i cloni sono uguali in tutto e per tutto alle versioni di Arduino, quindi se avete acquistato (per esempio) una scheda Elegoo Mega 2560 R3 potete tranquillamente scegliere la configurazione della scheda Arduino Mega 2560 R3 con la garanzia che funzionerà esattamente

NOTA

Arduino è un progetto open source ormai noto a molti appassionati di elettronica, ma non è ancora del tutto nota la possibilità di gestire lo sviluppo degli sketch attraverso l'IDE di Visual Studio, utilizzando l'add-in Arduino IDE for Visual Studio. Questo add-in richiede sempre l'installazione dell'Arduino IDE originale, ma lo utilizza dietro le quinte, esponendo i servizi attraverso il celebre ambiente di sviluppo di Microsoft.

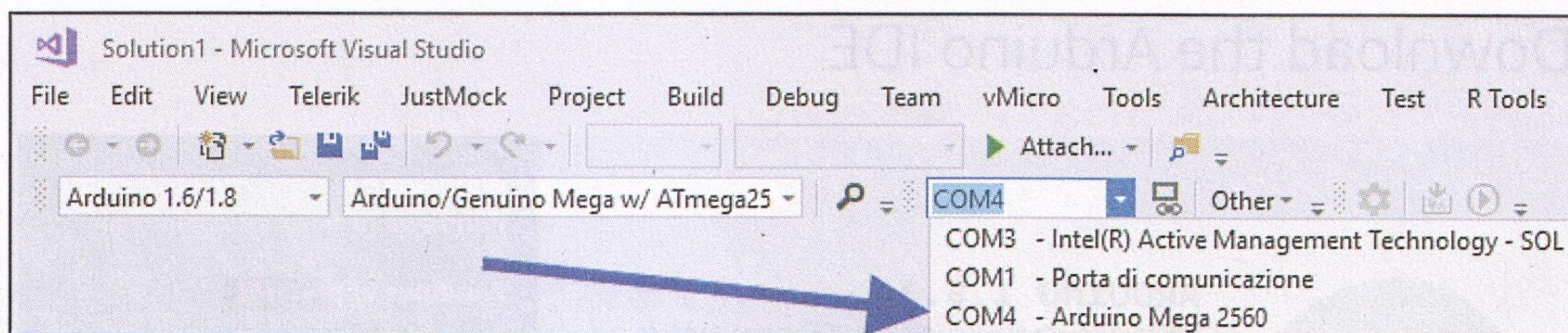



Fig. 6: L'elenco delle porte COM con evidenziata quella connessa ad Arduino.

come l'originale. Noterete che in Visual Studio sono apparse anche delle nuove barre di pulsanti denominate *Micro Boards* (per la scelta dell'IDE e della scheda), *Micro Serial Communications* (per la scelta della porta di comunicazione) e *Micro Project* (per compilare il progetto e inviarlo alla scheda). Proprio nella barra *Micro Serial Communications* dobbiamo scegliere la porta di connessione con la scheda Arduino, in modo da poter comunicare con la stessa (altrimenti perché avremmo fatto tutta questa fatica finora?). Per sapere quale porta utilizzare, però, dobbiamo prima connettere fisicamente la scheda a una porta USB del computer, perché è nel momento in cui inseriamo il connettore che avviene la magia.

NOTA

Con Visual Studio è anche possibile creare un'applicazione, in linguaggio Visual Basic, in grado di interfacciarsi con una scheda Arduino: basta saper gestire il dialogo con la scheda attraverso la porta seriale costituita dalle classiche porte USB (viste dal sistema come porte COM). In questo articolo vediamo come si configura e gestisce la comunicazione via USB con la scheda Arduino.

ARDUINO INCONTRA IL PC

Prendete la scheda Arduino che volete collegare, inserite l'estremità del cavo USB nell'apposita presa (**fig. 5**) e poi inserite l'altra estremità su una porta USB del computer.

La scheda viene immediatamente riconosciuta dal PC e vengono installati gli opportuni driver. Se state lavorando con Windows 10, probabilmente nell'angolo inferiore destro dello schermo vi sarà apparsa una notifica con

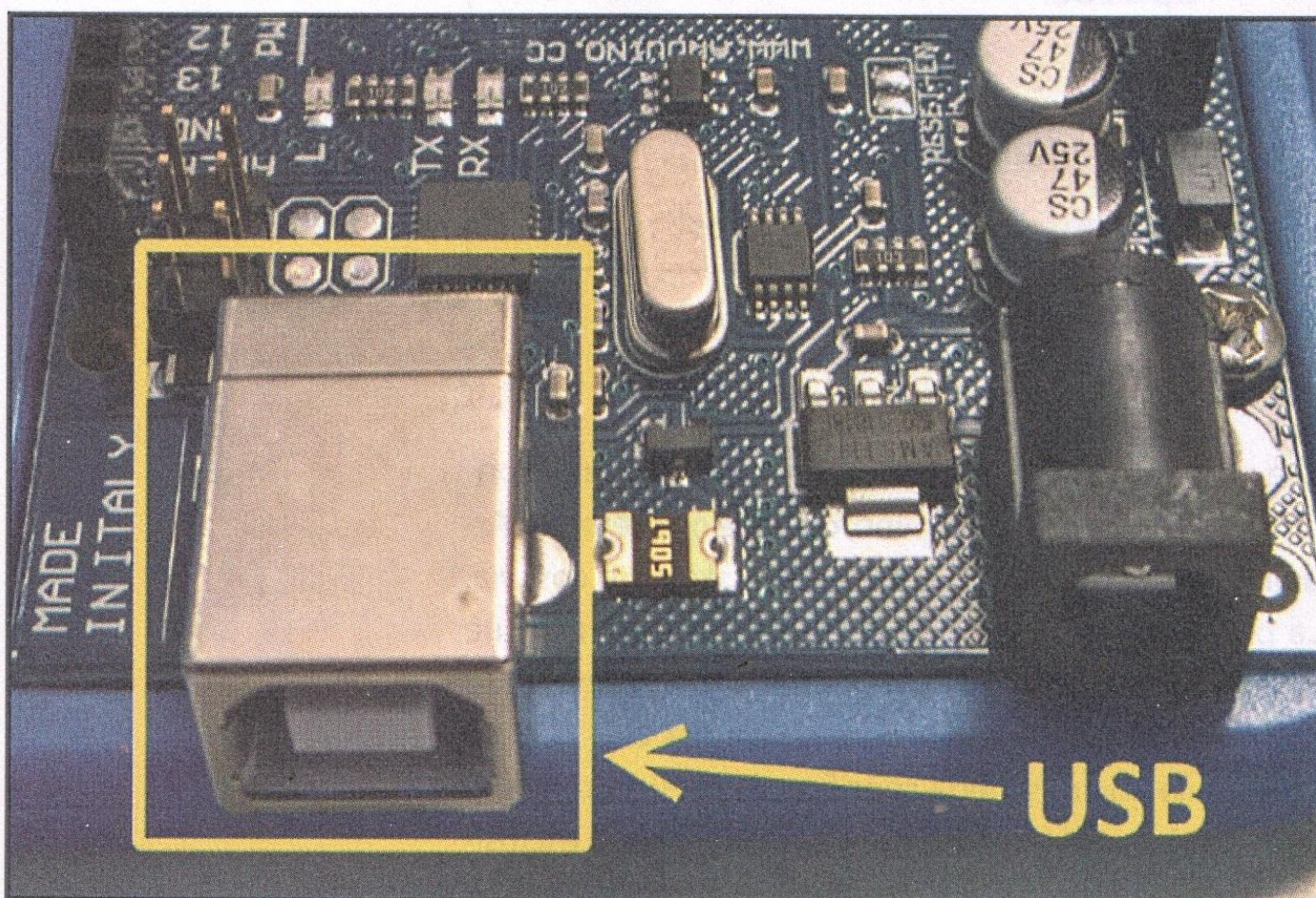



Fig. 5: La presa USB della scheda Mega 2560 R3.

l'indicazione della porta di comunicazione utilizzata. Se il messaggio vi è sfuggito, niente paura, perché potete recuperare il nome della porta in due modi molto semplici: nella barra dei pulsanti *Micro Serial Communications* cliccate sulla casella a discesa. Vedrete elencate tutte le porte COM, con evidenziata quella a cui abbiamo connesso Arduino (**fig. 6**).

Possiamo fare la stessa cosa anche con Arduino IDE, al di fuori di Visual Studio, anzi, vi consigliamo di procedere anche qui con la configurazione, per poter lavorare anche al di fuori di Visual Studio, se necessario. In questo caso dobbiamo selezionare la voce di menu *Strumenti > Porta*. Anche in questo caso la porta connessa ad Arduino è evidenziata dal nome della scheda. Una volta impostata la porta corretta abbiamo completato tutto il lavoro di preparazione dell'ambiente di lavoro e non ci resta che fare un test.

TEST DI COMUNICAZIONE TRA VISUAL STUDIO E ARDUINO

Per verificare che sia veramente tutto a posto possiamo fare un semplice test che ci permetta di capire se siamo in grado di aprire uno sketch, compilarlo, inviarlo alla scheda ed eseguirlo correttamente. Il modo più facile per fare questa prova è quello di far lampeggiare il LED presente sulla scheda, connesso anche al PIN 13, anche se per questo test non useremo i PIN per una questione di semplicità. Visto, però, che la scheda solitamente ha già precaricato lo sketch Blinker che fa lampeggiare il LED (un secondo acceso, un secondo spento), per vedere un cambiamento nel comportamento della scheda dobbiamo modificare i tempi di accensione e di spegnimento. Ecco quindi la sequenza di operazioni che vi consigliamo di seguire:

1. con la voce di menu *vMicro > Visual Micro Explorer*, aprite la finestra dell'omonimo strumento;
2. selezionate la scheda *Examples* e poi aprite il nodo *Examples > 01.Basics*;
3. cliccate sulla voce di colore azzurro *Blink*: questa voce permette di aprire un progetto di Visual Studio con lo *sketch* opportunamente caricato; se invece scendete di un livello il nodo stesso e cliccate sulla voce verde *Blink.ino*, vedrete che non verrà creato un progetto ma verrà solamente aperto

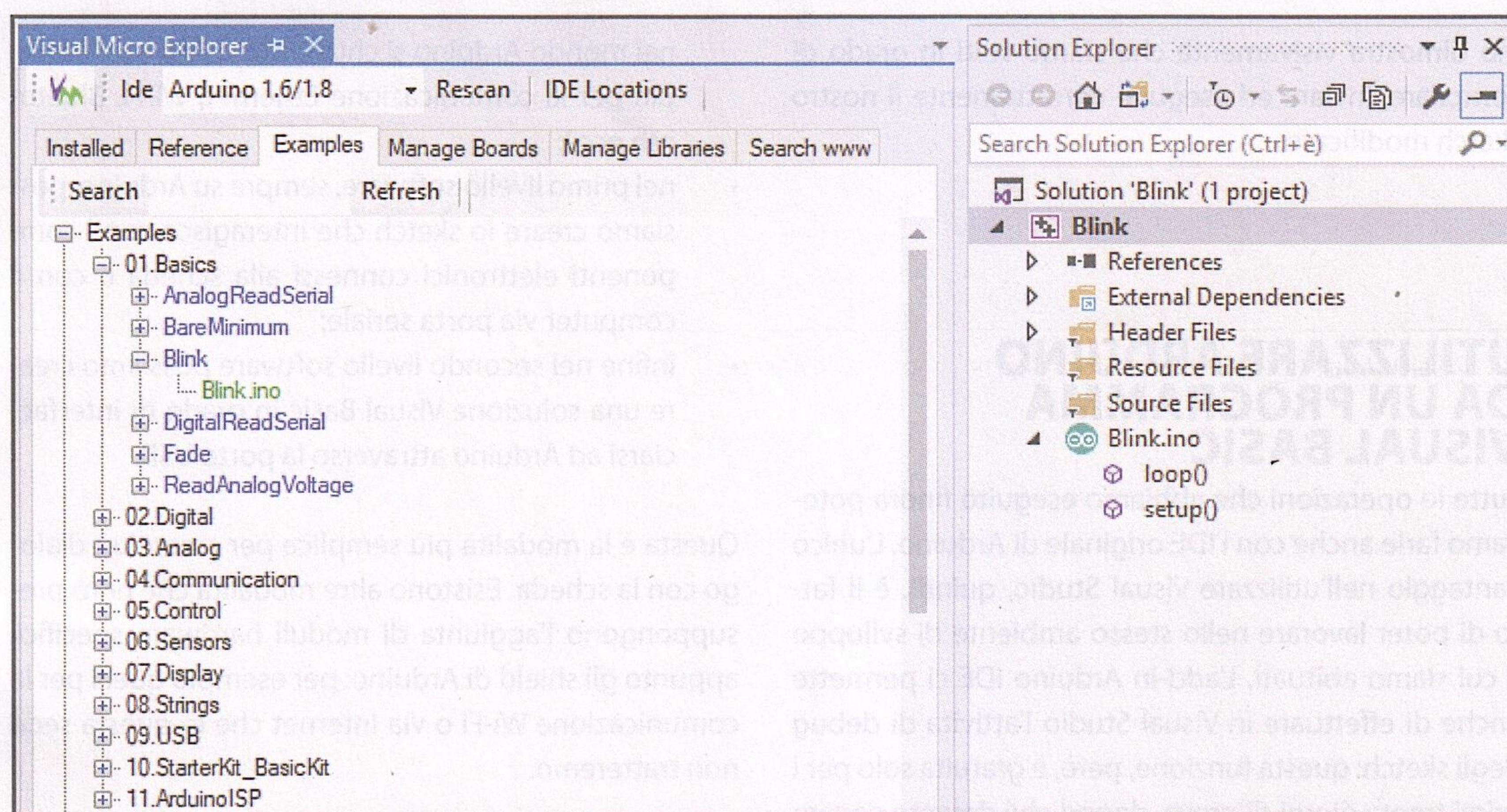
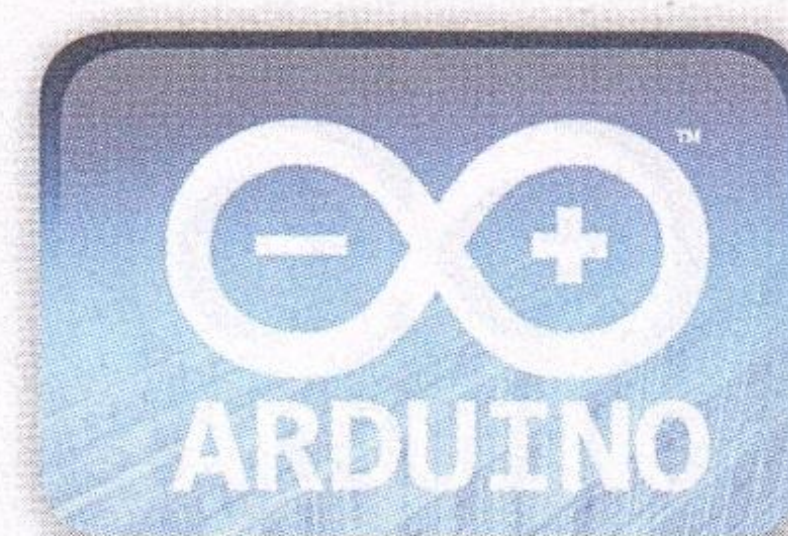



Fig. 7: A sinistra è visibile la finestra Visual Micro Explorer, mentre a destra c'è la finestra Solution Explorer con il progetto Blink aperto.

il file dello sketch, rendendo però impossibile la compilazione e l'upload dello stesso sulla scheda (**fig. 7**);

4. modificate lo sketch come segue:

```
// Progetto: Blink modificato
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}
void loop()
{
  digitalWrite(LED_BUILTIN, HIGH);
  delay(500);
  digitalWrite(LED_BUILTIN, LOW);
  delay(2000);
}
```

5. se volete solamente compilare il progetto cliccate sul pulsante *Build*, mentre se volete compilare e inviare alla scheda Arduino cliccante sul pulsante *Build and Upload*. I due pulsanti citati sono presenti nella barra dei pulsanti Micro Project.

Non appena avrete inviato il codice compilato alla scheda Arduino, questa immediatamente attiverà il nuovo programma. Rispetto al codice standard dello sketch Blink che prevede l'accensione del LED per un secondo (1.000 millisecondi) e lo spegnimento del LED per un altro secondo, nello sketch modificato abbiamo impostato mezzo secondo di accensione e due secondi di spegnimento. Il comportamento della scheda Ardu-

Alla scheda Arduino possiamo collegare qualsiasi componente elettronico e anche dei moduli aggiuntivi (che nel mondo Arduino si chiamano shields) preconfezionati che hanno le possibilità più disparate: comunicazioni via Ethernet, Bluetooth o Wi-Fi, ma anche sensori per qualsiasi grandezza fisica che può essere misurata. Attraverso Arduino, tutti questi dispositivi sono messi a disposizione di un'applicazione Visual Basic, permettendo così di realizzare qualcosa di unico.

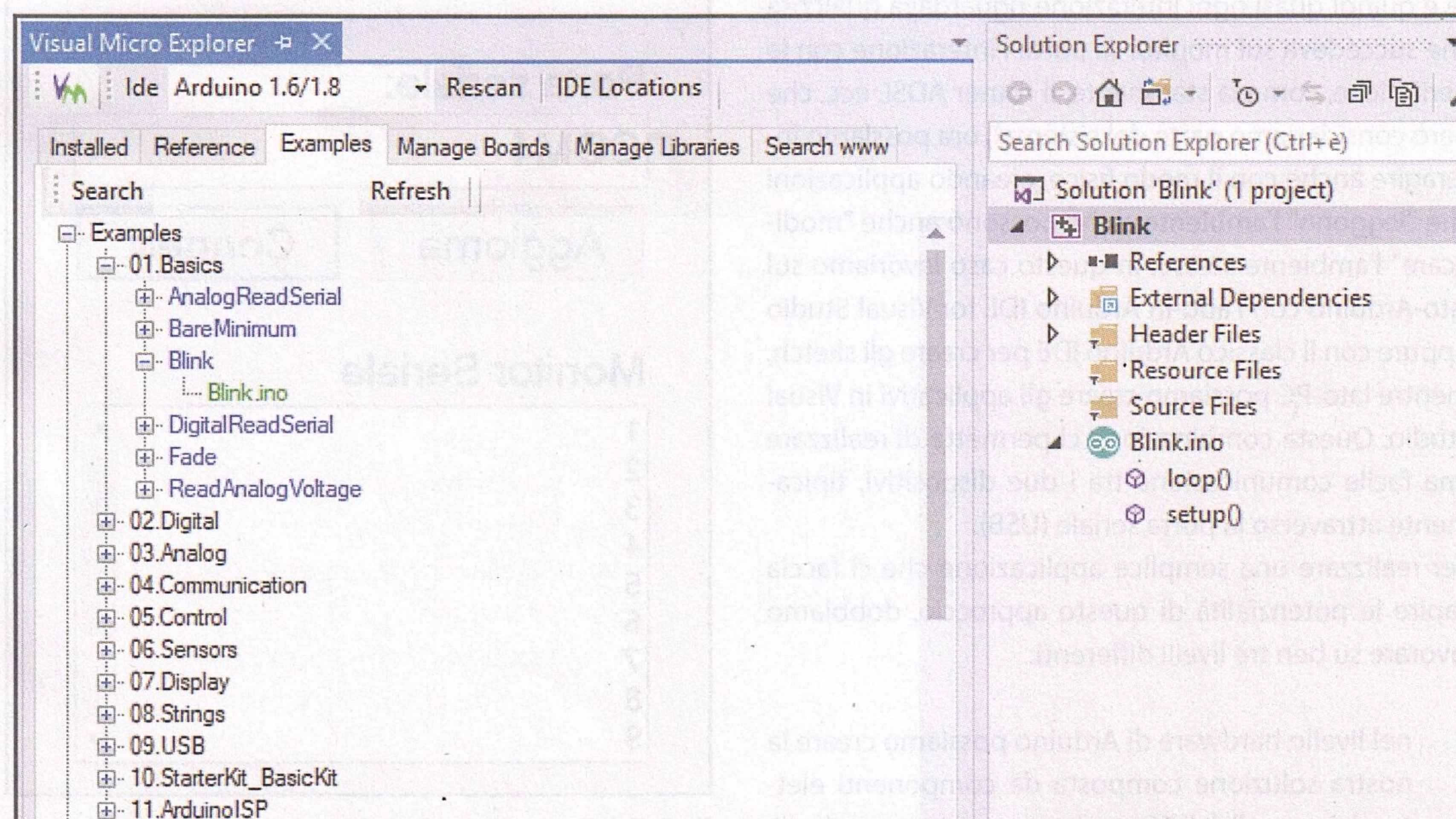



Fig. 8: La finestra per l'attivazione delle funzionalità aggiuntive di debug o per l'attivazione della trial.

ino dimostra visivamente che siamo stati in grado di compilare, inviare ed eseguire correttamente il nostro sketch modificato.

UTILIZZARE ARDUINO DA UN PROGRAMMA VISUAL BASIC

Tutte le operazioni che abbiamo eseguito finora potevamo farle anche con l'IDE originale di Arduino. L'unico vantaggio nell'utilizzare Visual Studio, quindi, è il fatto di poter lavorare nello stesso ambiente di sviluppo a cui siamo abituati. L'add-in Arduino IDE ci permette anche di effettuare in Visual Studio l'attività di debug degli sketch: questa funzione, però, è gratuita solo per i primi trenta giorni di prova, dopodiché dovrete pagare la licenza (con un costo che varia da 45 $ a 200 $, in base al profilo commerciale o no-profit). In caso contrario le funzioni aggiuntive cesseranno di funzionare e rimarranno le funzioni di base. Per attivare il periodo di prova gratuito potete selezionare il menu *vMicro > Buy, Activate Or Start A Trial*. Apparirà una finestra simile alla **fig. 8**.

Per avviare solo il periodo di prova potete cliccare su *Start trial*, per acquistare la licenza potete cliccare su **Buy safely on-line** (questo pulsante apre l'apposita pagina di sottoscrizione sul sito del produttore), oppure per attivare la chiave che avete acquistato potete cliccare appunto su *Activate purchased key*. Il vero salto di qualità lo otteniamo unendo la potenza del PC, attraverso una nostra applicazione Visual Basic, e la potenza di Arduino, con i suoi sketch e la possibilità di connettere il microcontroller ad altri componenti elettronici e a innumerevoli sensori. In sostanza, mentre finora eravamo abituati a lavorare in Visual Studio solo con il software e quindi quasi ogni interazione riguardava qualcosa che succedeva sul monitor (a parte l'interazione con le periferiche, come la stampante, il router ADSL ecc. che però consideriamo parte del sistema), ora possiamo interagire anche con il modo fisico, creando applicazioni che "leggono" l'ambiente e che possono anche "modificare" l'ambiente stesso. In questo caso lavoriamo sul lato-Arduino con l'add-in Arduino IDE for Visual Studio oppure con il classico Arduino IDE per creare gli sketch, mentre lato-PC possiamo creare gli applicativi in Visual Studio. Questa combinazione ci permette di realizzare una facile comunicazione tra i due dispositivi, tipicamente attraverso la porta seriale (USB).

Per realizzare una semplice applicazione che ci faccia capire le potenzialità di questo approccio, dobbiamo lavorare su ben tre livelli differenti:

- nel livello hardware di Arduino possiamo creare la nostra soluzione composta da componenti elettronici semplici (LED, resistenze, relay, ponticelli, sensori ecc.) o da circuiti integrati già pronti che nel mondo Arduino si chiamano shield (per esempio per la comunicazione Ethernet, Wi-Fi, Bluetooth ecc.);
- nel primo livello software, sempre su Arduino, possiamo creare lo sketch che interagisce con i componenti elettronici connessi alla scheda e con il computer via porta seriale;
- infine nel secondo livello software possiamo creare una soluzione Visual Basic in grado di interfacciarsi ad Arduino attraverso la porta USB.

Questa è la modalità più semplice per creare un dialogo con la scheda. Esistono altre modalità che però presuppongono l'aggiunta di moduli hardware specifici, appunto gli shield di Arduino: per esempio quelli per la comunicazione Wi-Fi o via Internet che in questa sede non tratteremo.

NOTA

Gli sketch, come sarà noto a molti, sono programmi scritti in un linguaggio C semplificato e adattato alle schede Arduino. La sintassi è piuttosto facile per chi "mastica" linguaggi come C, C# o JAVA ed è facilmente assimilabile anche da uno sviluppatore Visual Basic. In ogni caso esistono delle guide o dei tutorial in Internet che agevolano il percorso didattico: in una giornata potete diventare esperti del linguaggio di Arduino.

LAVORARE CON IL SOFTWARE

Per il momento ci limitiamo a un progetto "leggero" basato solo sul lato software di Arduino e del PC, in modo da prendere confidenza con la comunicazione via porta seriale.

In seguito estenderemo il progetto con ulteriori funzionalità, mettendo in gioco anche l'aspetto più hardware. Per prima cosa creiamo uno sketch che semplicemente invia un numero crescente sulla porta seriale e fa accendere il LED ogni dieci secondi per la durata di un secondo:

```
// == Progetto: Arduino_SerialPort_1 ==
int conta = 0;
```

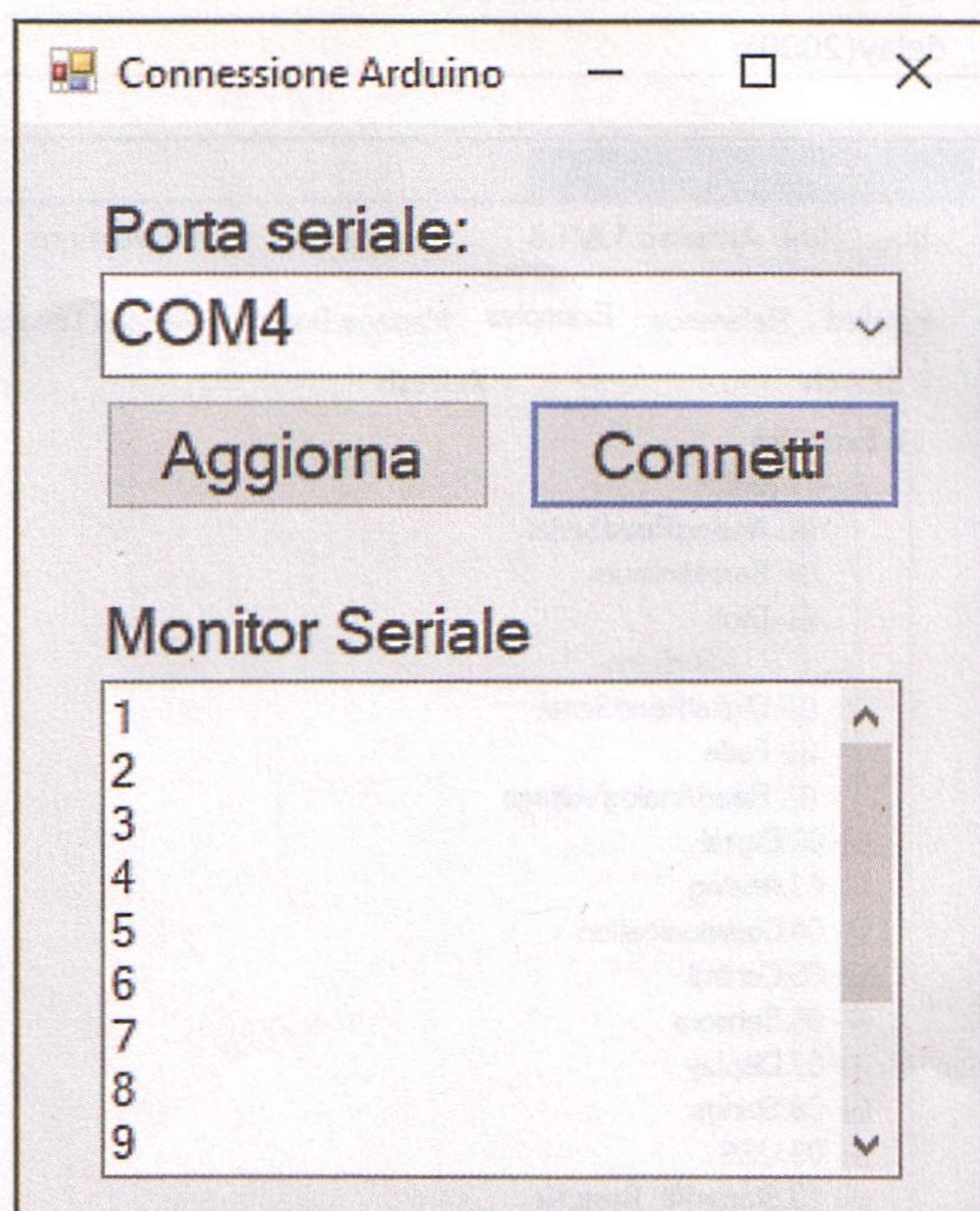



Fig. 9: L'applicazione Visual Basic denominata Arduino_SerialPort_VB_1.

```
int contaLED = 0;
byte LED = 13;

void setup() {
  pinMode(LED, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  conta++;
  contaLED++;
  Serial.println(conta);
  if (contaLED == 10)
  {
    contaLED = 0;
    digitalWrite(LED, HIGH);
  }
  delay(1000);
  digitalWrite(LED, LOW);
}
```

Collegate Arduino, compilate e caricate lo sketch e vedrete lampeggiare ogni secondo il LED TX che indica la trasmissione di dati sulla seriale. Ogni dieci secondi, poi, si accenderà il LED collegato al PIN 13 spegnendosi dopo un secondo. Inoltre, se aprite il monitor seriale, vedrete apparire un numero crescente ogni secondo.
Ora procediamo con la creazione di un'applicazione Visual Basic (in Windows Forms per semplicità) in grado di leggere i dati provenienti dalla porta seriale.
Aggiungete al form due pulsanti, una ComboBox e una TextBox con attivata la proprietà MultiLine, disposti come nella **fig. 9**.
All'applicazione dobbiamo aggiungere anche altri due componenti: *SerialPort* e *Timer*. Questi componenti saranno gestiti direttamente dall'applicazione Visual Basic.

```
' == Progetto Arduino_SerialPort_VB_1 ==
Imports System.IO.Ports

Public Class Form1
  Dim comPort As String = ""
  Dim dataRx As String = ""
  Dim flagConnesso As Boolean = False

  Public Sub New()

    ' This call is required by the designer.
    InitializeComponent()

    ' Add any initialization after the
                    InitializeComponent() call.
    Timer1.Enabled = False
    Timer1.Interval = 500
    Call COMrefresh()
  End Sub

Private Sub btnRefresh_Click(sender As Object, e As
                    EventArgs) Handles btnRefresh.Click
  Call COMrefresh()
End Sub

Private Sub btnConnect_Click(sender As Object, e As
                    EventArgs) Handles btnConnect.Click
  If flagConnesso = False Then
    If (comPort <> "") Then
      SerialPort1.Close()
      SerialPort1.PortName = comPort
      SerialPort1.BaudRate = 9600
      SerialPort1.DataBits = 8
      SerialPort1.Parity = Parity.None
      SerialPort1.StopBits = StopBits.One
      SerialPort1.Handshake = Handshake.None
      SerialPort1.Encoding = System.Text.Encoding.
                    Default
      SerialPort1.ReadTimeout = 10000
      SerialPort1.Open()
      flagConnesso = True
      btnConnect.Text = "Disconnetti"
      Timer1.Enabled = True
    Else
      MessageBox.Show("Seleziona una porta COM!")
    End If
  Else
    Timer1.Enabled = False
    SerialPort1.Close()
    btnConnect.Text = "Connetti"
  End If
End Sub

Public Sub COMrefresh()
  cmbCOMports.Items.Clear()
  For Each serialPort As String In My.Computer.Ports.
                    SerialPortNames
    cmbCOMports.Items.Add(serialPort)
  Next
End Sub

Private Sub cmbCOMports_
        SelectedIndexChanged(sender As Object,
        e As EventArgs) Handles cmbCOMports.
                    SelectedIndexChanged
  If (cmbCOMports.SelectedItem <> "") Then
    comPort = cmbCOMports.SelectedItem
  End If
End Sub

Private Sub Timer1_Tick(sender As Object, e As
                    EventArgs) Handles Timer1.Tick
  TextBox1.Text &= RxSerialData()
End Sub

Function RxSerialData() As String
  Dim text As String
  Try
```

Lavorando con l'hardware avrete a che fare con LED, resistenze, ponticelli (cioè cavetti), linee di tensione, condensatori, schede e molto altro. Qualsiasi modifica ai circuiti, per aggiungere, togliere o modificare la posizione dei componenti elettronici, deve essere fatta con la scheda scollegata da qualsiasi tipo di alimentazione (cavo USB, alimentatore e batteria). In caso contrario non solo rischiate di danneggiare Arduino, ma anche voi stessi!

```
            text = SerialPort1.ReadExisting
            If text Is Nothing Then
                Return "nothing" & vbCrLf
            Else
                Return text
            End If
        Catch ex As TimeoutException
            Return "Errore: timeout in lettura della porta
                                                seriale."
        End Try
    End Function
End Class
```

NOTA

Un progetto formato da hardware di Arduino e altri componenti, da software di Arduino (sketch) e da software su PC (per esempio un'applicazione Visual Studio) può essere particolarmente complesso. Ricordate che la scheda Arduino è sequenziale, mentre il PC è multitasking, quindi è facile perdere il controllo della situazione. Fate sempre test approfonditi per evitare errori di sincronizzazione tra le varie componenti che potrebbero compromettere tutto il progetto.

Dopo aver selezionato la porta di comunicazione opportuna, cliccate sul pulsante *Connetti*. Se avete il dubbio di quale sia quella corretta, scollegate il cavo USB dalla scheda arduino, aggiornate l'elenco delle porte COM, ricollegate la scheda e aggiornate di nuovo: la porta COM che è apparsa nell'elenco è quella da utilizzare. Appena avrete aperto la connessione, vedrete che nella casella di testo appariranno uno alla volta dei numeri progressivi, analogamente al Monitor Seriale dell'applicazione Arduino IDE. Inoltre, vedrete lampeggiare ancora il LED della scheda e il LED TX con i tempi che abbiamo impostato nello sketch.

LAVORARE CON L'HARDWARE

Lavorando a un progetto che combina una parte di hardware (la scheda Arduino con uno o più componenti elettronici), una parte di software della scheda Arduino e una parte di software dell'applicazione Visual Studio le cose possono complicarsi un po'. Quanto meno è opportuno fare un po' di sperimentazione in modo da evitare strani errori o false interpretazioni. Per prima cosa consideriamo che Arduino non è multitasking e quindi esegue le istruzioni in modo sequenziale. Tuttavia la parte hardware si può inserire nei processi e modificare il normale flusso del programma (sia quando di tali variazioni ne teniamo conto nello sketch che scriviamo, sia quando decidiamo di gestire i segnali di interrupt). Dal lato dell'applicativo Visual Studio, d'altro canto, dobbiamo tenere conto che mentre stiamo elaborando dei dati che ci sono arrivati dalla porta di comunicazione connessa con Arduino, il tempo continua a scorrere e Arduino sta già avanzando alle prossime istruzioni dello sketch. Nel prossimo esempio vediamo cosa succede quando, dopo aver ripreso tra le mani il primo semplice progetto, decidiamo di introdurre un pulsante collegato alla scheda Arduino. Di questo pulsante dobbiamo tenere conto sia nello sketch, sia nell'applicazione Visual Basic. Per prima cosa assicuriamoci di avere scollegato l'alimentazione di Arduino (mai lavorare sotto tensione… la sicurezza prima di tutto!) e poi inseriamo un pulsante e due ponticelli come nella **fig. 10.** La figura di sinistra rappresenta una scheda Arduino UNO con una piccola breadboard, mentre a destra potete vedere una scheda Elegoo Mega 2560 R3 (per questo progetto potete utilizzare qualsiasi versione di Arduino o di clone equivalente). Sulla breadboard abbiamo inserito un pulsante a cavallo del separatore e due ponticelli, uno blu e uno rosso, collegati rispettivamente a massa (GND) al PIN 8. Ora modifichiamo lo

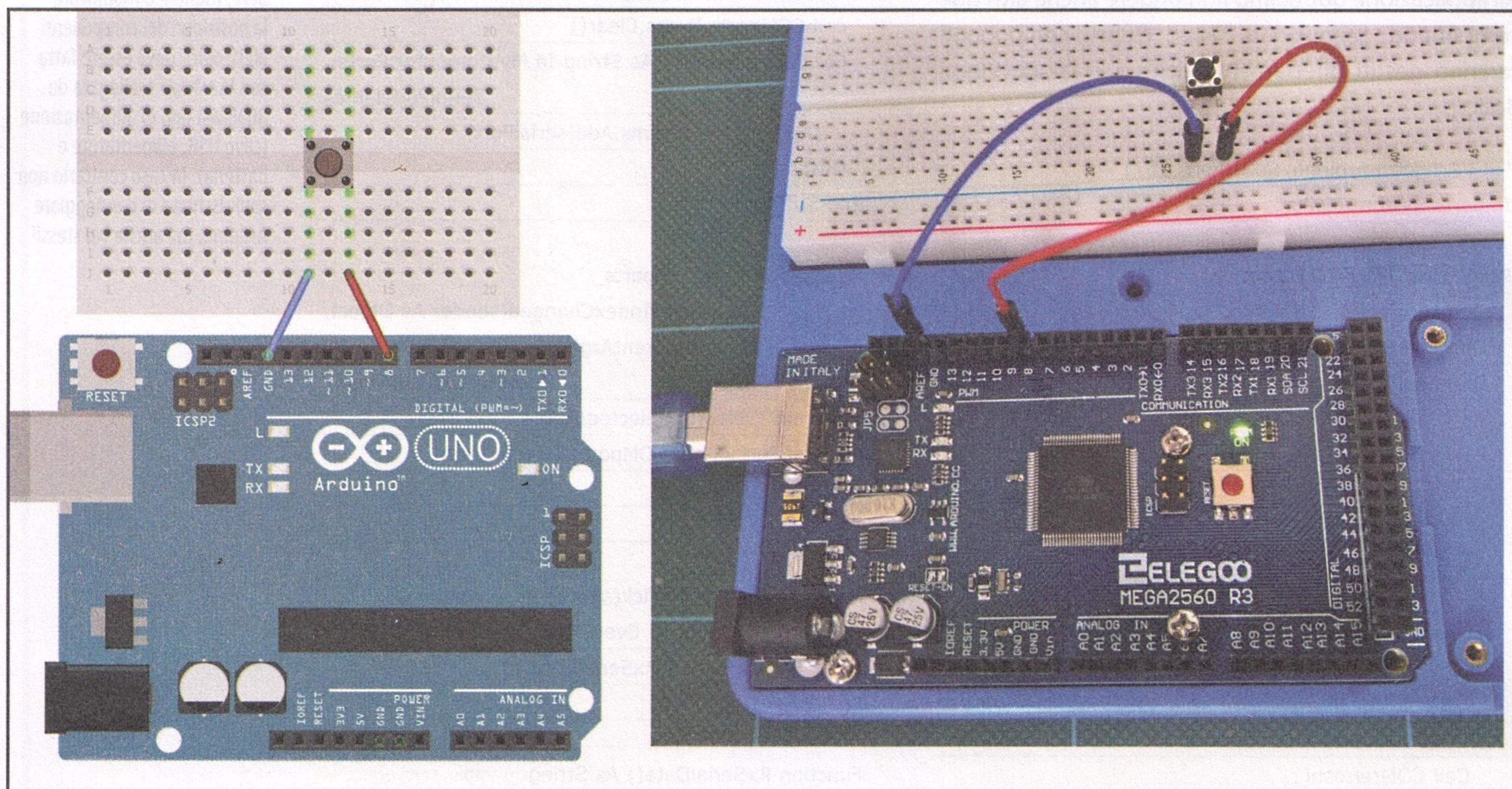



Fig. 10: Un semplice circuito collegato a una scheda Arduino (a sinistra lo schema creao con il software Fritzing e a destra la foto del circuito realizzato con un altro modello di scheda).

sketch come segue:

```
// == Progetto: Arduino_SerialPort_2 ==
int conta = 0;
int button = 8;

void setup() {
  pinMode(13,OUTPUT);
  pinMode(button, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop() {
  digitalWrite(13,LOW);
  conta++;
  Serial.println(conta);
  if(digitalRead(button)==LOW)
  {
   Serial.println("<P>");
  }
  delay(1000);
}
```

Come potete vedere abbiamo aggiunto la definizione del pulsante (*INPUT_PULLUP*) il cui segnale è collegato al PIN 8. Abbiamo anche spento il LED della scheda abbassando il livello sul PIN 13, per evitare di lasciarla accesa, e abbiamo tolto tutte le operazioni di lampeggiamento del LED perché ora ci interessa seguire di più il funzionamento del pulsante. Se premete il pulsante, viene rilevato un livello basso (*LOW*) e quindi viene inviata sulla seriale la stringa "<P>". Ovviamente, finché tenete premuto il pulsante il livello resta basso e quindi vengono generate più stringhe "<P>" in sequenza. Vediamo ora cosa succede sul lato PC con la nostra applicazione Visual Basic:

```
' == Progetto Arduino_SerialPort_VB_2 ==
  Private Sub Timer1_Tick(sender As Object, e As
                    EventArgs) Handles Timer1.Tick
   Dim testo As String = ""
   testo = RxSerialData()
   If testo.Contains("<P>") Then
     MessageBox.Show("OK")
     testo = testo & " - HAI PREMUTO IL PULSANTE!"
                                         & vbCrLf
   End If
   TextBox1.Text = testo & TextBox1.Text
  End Sub

  Function RxSerialData() As String
   Dim text As String
   Try
     text = SerialPort1.ReadExisting
     If (text = "") Or (text Is Nothing) Then
       Return ""
     Else
       Return text
     End If
   Catch ex As TimeoutException
     Return "Errore: timeout in lettura della porta
                              seriale." & vbCrLf
   End Try
  End Function
```

Quando riceviamo la stringa "<P>" accodiamo anche la stringa "HAI PREMUTO IL PULSANTE!" per rendere più evidente l'azione sul pulsante, ma appena prima abbiamo anche inserito una istruzione per visualizzare un messaggio attraverso un MessageBox. Vedrete che cliccando subito sul messaggio "OK", la sequenza andrà avanti nell'ordine corretto. Se però non cliccate subito il pulsante e lo lasciate in attesa, Arduino andrà comunque avanti nella produzione del numero progressivo e nell'invio del dato alla seriale. Inoltre, cliccando ancora sul pulsante collegato alla scheda, appariranno più MessageBox in attesa del vostro click. Cliccando, quindi, sugli "OK" di queste finestre, vedrete che nel Monitor Seriale appariranno i numeri che dovevano apparire in precedenza, ma fuori sequenza (Fig. 11). In sostanza, le chiamate alla funzione di visualizzazione dei messaggi (o meglio, le chiamate all'evento Timer1.Tick) sono rimaste in sospeso finché non le avete liberate, completando le istruzioni rimanenti. Quest'ultimo esempio dimostra che l'effettiva sincronizzazione delle due piattaforme hardware, con i relativi software, deve essere testata con attenzione per evitare di creare una situazione non completamente corretta o addirittura errata.

Mario De Ghetto

Fig. 11: L'applicazione Visual Basic del secondo esempio, con l'errore di sequenza dovuto alla visualizzazione di un messaggio che sospende le istruzioni successive.

L'AUTORE

Mario De Ghetto è un appassionato di informatica a 360 gradi, ma soprattutto di programmazione. Si è laureato in ingegneria informatica all'Università di Padova, è Microsoft MVP dal 2008, è autore di vari libri come "Visual Basic 2010 spiegato a mia nonna" e "SQL Server 2016 – La guida" e collabora da diversi anni con ioProgrammo. Può essere contattato tramite il blog http://deghetto.wordpress.com.

PROCESSING 3.3.5

Un ambiente di programmazione e un linguaggio davvero peculiare, rivolto principalmente ai progettisti impegnati nelle interfacce grafiche e agli artisti visuali che utilizzano la computer graphic come mezzo espressivo. Processing può essere visto anche come uno strumento per apprendere i principi della programmazione attraverso un approccio visuale ed "artistico".

JCPP EDIT 3.8

Un IDE che fa della semplicità la sua stella polare: sarà possibile creare progetti in Java, C, C++, JavaScript, XML e HTML. L'interfaccia ci aiuta in tutte le fasi della scrittura del codice in maniera proattiva.

APTANA STUDIO 3.6

Sviluppare e testare l'intera applicazione web utilizzando un unico ambiente grazie a un tool che permette di interagire perfettamente con HTML5, JavaScript, Ruby, Rails, PHP e Python.

CODELOBSTER PHP EDITION 5.13

Un IDE per PHP gratuito e ricco di funzionalità: evidenziazione sintattica, autocompletamento, help contestuale... Abbiamo tutto quello che si richiede ad un editor moderno, assieme a un efficace debugger, un SQL manager ed il pieno supporto per il trasferimento dei file via FTP.

BLUEGRIFFON 2.3.1

Un Web editor, open source, WYSIWYG, compatibile con gli standard HTML 4.0, HTML5, XHTML 1.0 e XHTML5. BlueGriffon è stato scritto da Daniel Glazman, lo sviluppatore principale di Nvu, basandosi sull'engine di Firefox.

PMD 5.8.1

Un analizzatore di codice sorgente che permette di alleggerire e migliorare i sorgenti, attraverso un'approfondita ricerca degli errori più o meno comuni che caratterizzano la scrittura del codice: variabili inutilizzate, inutile creazione di oggetti e così via.

COMPACT disc
www.ioprogrammo.it
EDIZIONI MASTER
ioProgrammo n.218
I MIGLIORI STRUMENTI DI SVILUPPO

PROCESSING 3.3.5
La più diffusa piattaforma per programmare esperienze visuali
- Interfacce 3D
- Visual art
- Modellazione interattiva

JCpp Edit 3.8
Un IDE per Java e C++ che rende più elegante e leggibile il tuo codice

Aptana Studio 3.6.1
Un editor per HTML5, JavaScript e CSS3, Ruby, Rails, PHP e Python

CodeLobster PHP 5.13
Un ambiente integrato per lo sviluppo e il debug di PHP

PMD 5.8.1
Analizza e migliora il codice sorgente dei tuoi progetti software

PMD
Codelobster PHP Edition
Aptana Studio
JCpp Edit
Processing

I MIGLIORI STRUMENTI DI SVILUPPO
ioProgrammo
www.ioprogrammo.it
n.218